SICE関西支部主催のオンライン勉強会2020

Pythonで学ぶ制御工学

Part3: ループ整形と状態空間モデルを用いた設計

大阪大学
OSAKA UNIVERSITY

南 裕樹

勉強会の概要

流れや雰囲気を体感してください
独学のハードルを下げるのが目的です！本を読んでください！

Part 1

制御工学の基礎
- 制御とは，フィードバック制御，制御系設計

Pythonプログラミングの基礎
- Jupyter Notebook の使い方
- Pythonプログラミング　★ Python演習 1

制御系設計のためのモデル
- 伝達関数モデル，ブロック線図，時間応答，周波数応答　★ Python演習 2

Part 2

制御系設計のためのモデル（復習）
- 伝達関数モデル，時間応答，周波数応答

伝達関数モデルを用いた制御系設計
- 閉ループ系の設計仕様，PID制御，モデルマッチング　★ Python演習 3

Part 3

ループ整形による制御系設計
- 開ループ系の設計仕様，位相進み・遅れ補償　★ Python演習 4

状態空間モデルを用いた制御系設計
- 状態空間モデル，状態フィードバック　★ Python演習 5

今日の目標

Level☆☆★

- 開ループ系の設計仕様の用語を説明できる
- 開ループ系の安定余裕をPythonでチェックできる
- 状態空間モデルをPythonで記述できる

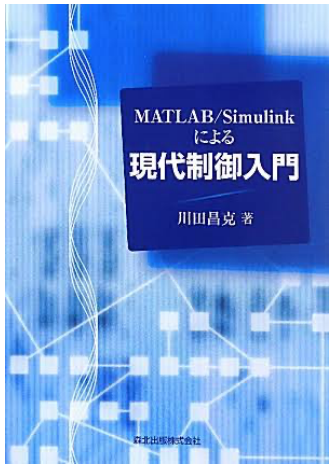
Level☆★★

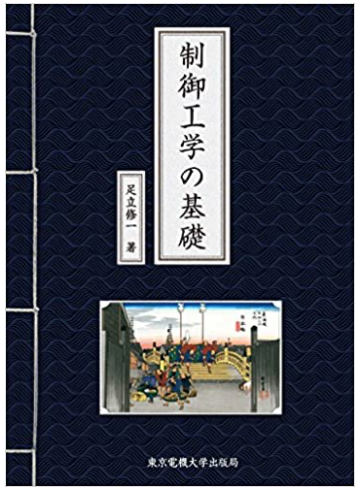
- ループ整形法の流れが理解できる
- 極配置法で状態フィードバックゲインが設計できる

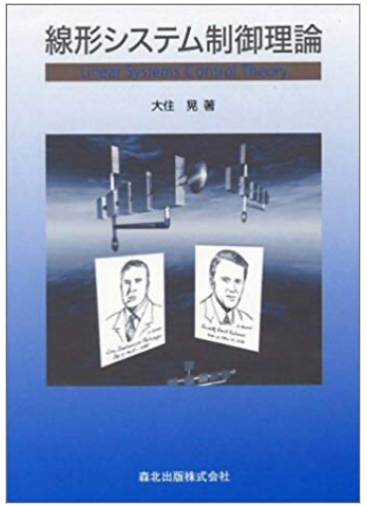
Level★★★

- 開ループ系の設計仕様を導くことができる
- 最適レギュレータやサーボ系，オブザーバを説明できる

参考図書







導入にもってこい

押さえておけば安心

数学的にしっかり勉強

補助資料の紹介

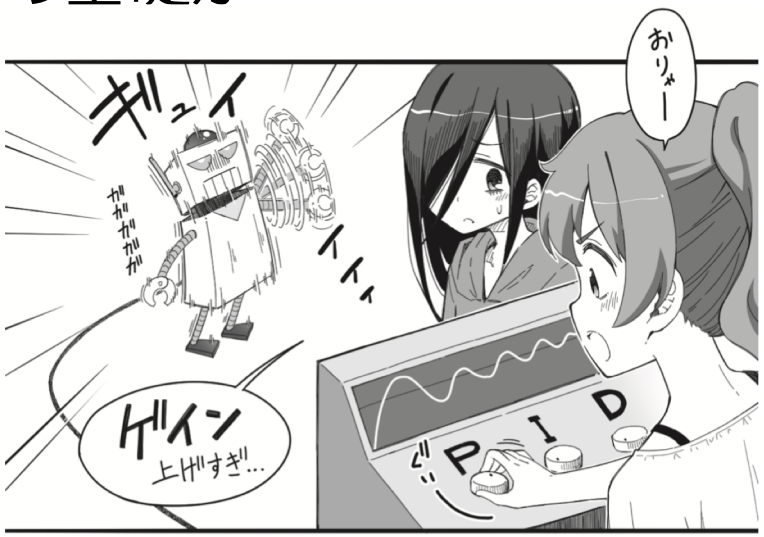
「Python × 制御工学」の日本初の本

古典制御，現代制御，ロバスト制御の基礎がギュッとつまっている

数学的な説明は少なめ

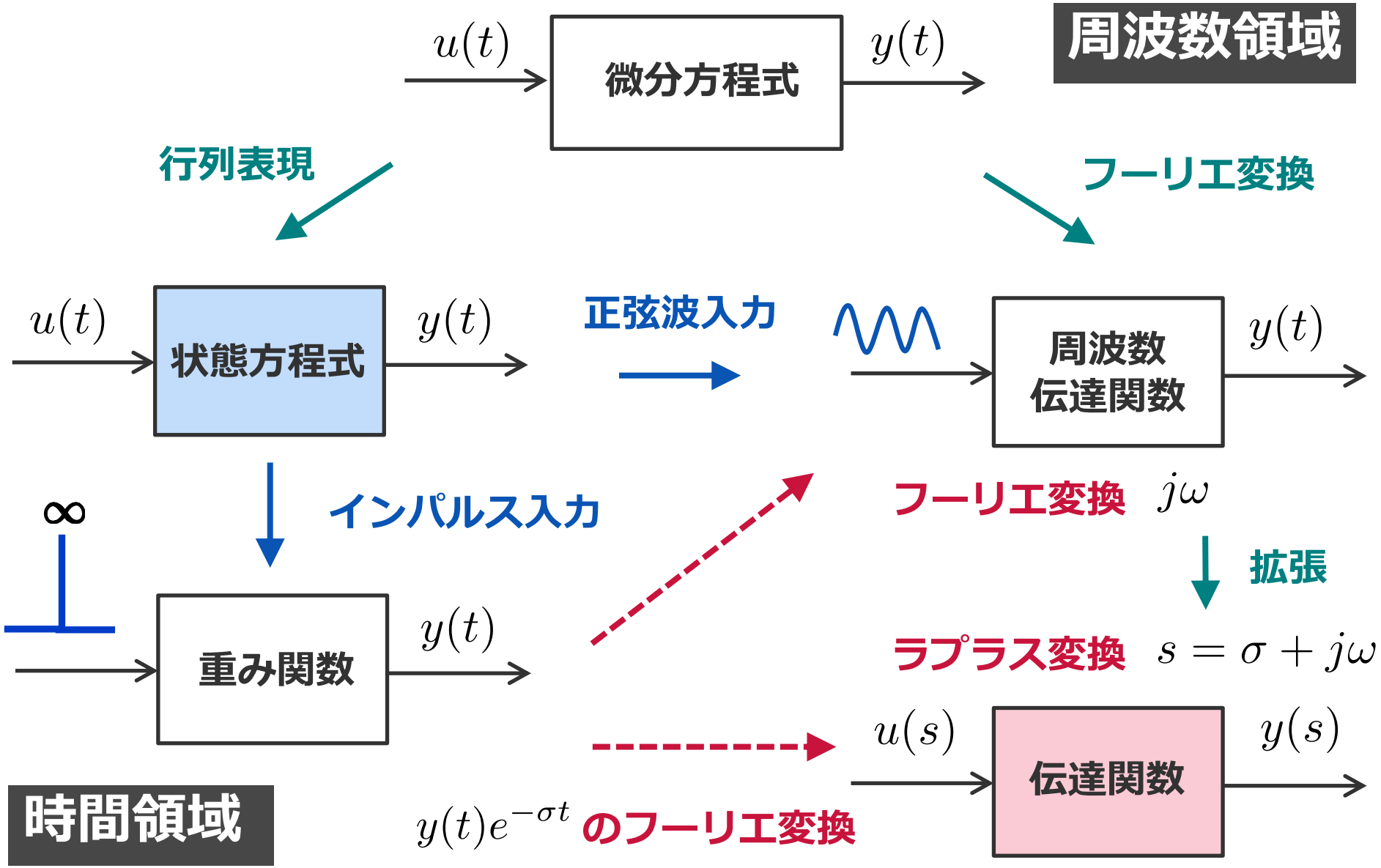
対話形式の説明やイラストを配置

サポートページには，
サンプルコードが公開されている
→ Python & MATLABコード



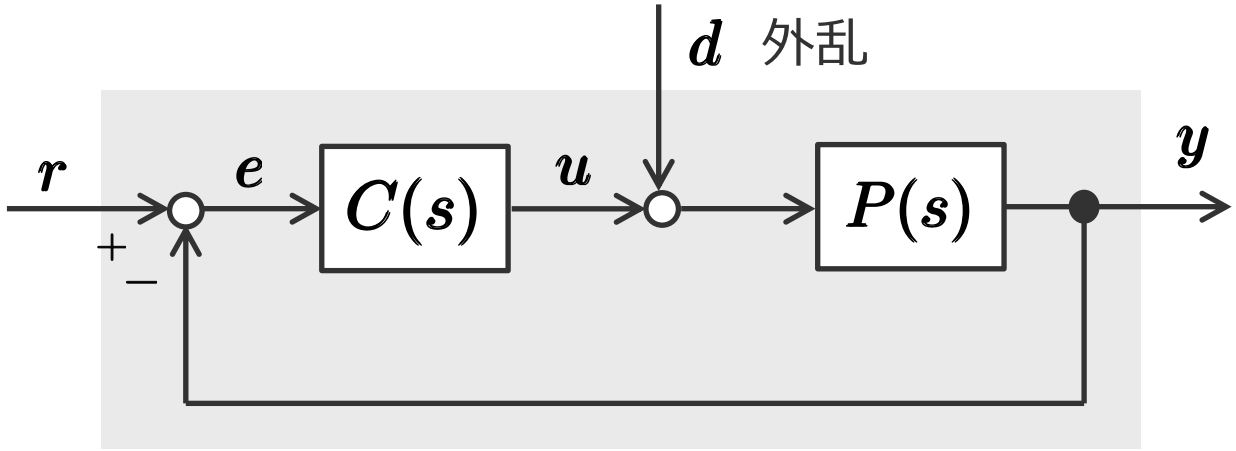
- 2,600 円＋税
- A5 272頁
- 2019/05/22 発行　Kindle版もあります

設計モデルの関係


$u(t)$
微分方程式
$y(t)$
周波数領域
行列表現
フーリエ変換
$u(t)$
状態方程式
$y(t)$
正弦波入力
周波数
伝達関数
$y(t)$
∞
インパルス入力
フーリエ変換 $j\omega$
拡張
重み関数
$y(t)$
ラプラス変換 $s = \sigma + j\omega$
$u(s)$
伝達関数
$y(s)$
時間領域
$y(t)e^{-\sigma t}$ のフーリエ変換

閉ループ系の設計仕様

フィードバック制御系を構築



閉ループ系の特性に注目

・どんな時間応答？
・どんな周波数応答？
・閉ループ極はどこに配置？

注目するポイント（設計仕様）

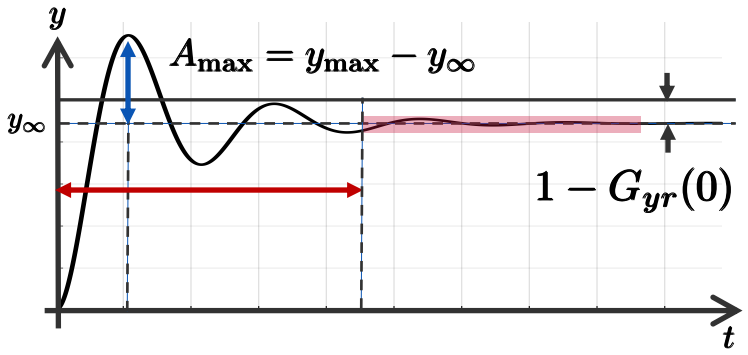
1）安定性

2）速応性 ＋ 減衰性

3）定常偏差

閉ループ系の設計仕様

閉ループ系の設計仕様



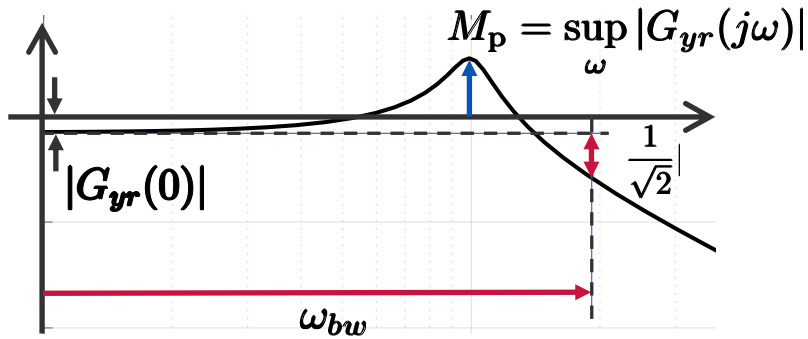
1）安定性 ➡ 内部安定

2）速応性，減衰性（≒ロバスト安定性）

- ➡ G_{yr} のステップ応答において整定時間が小さい
- ➡ 行き過ぎ量（オーバーシュート）が小さい
- ➡ G_{yr} のゲイン線図：バンド幅が十分大きい
- ➡ ピークゲイン $M_{\mathrm{p}} = \sup_{\omega} |G_{yr}(j\omega)|$ が小さい

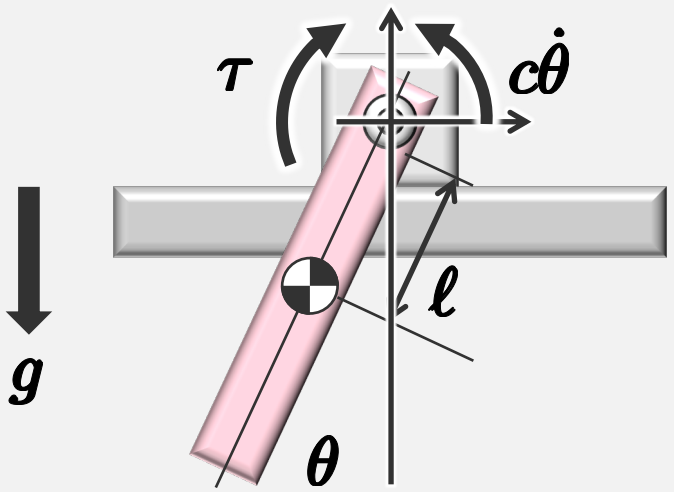
3）定常偏差

- ➡ $|G_{yr}(0)| = 1$



制御系設計の例題

２次遅れ系（アーム，RCL回路）が対象



$$P(s) = \frac{1}{Js^2 + cs + mg\ell}$$

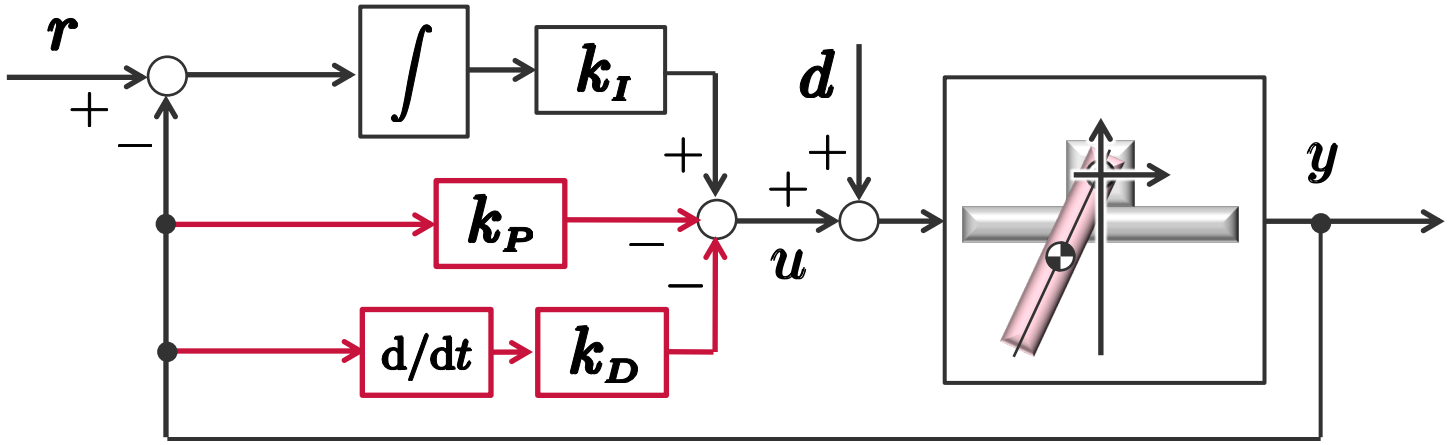
```
g = 9.81      # 重力加速度[m/s^2]
l = 0.2       # アームの長さ[m]
m = 0.5       # アームの質量[kg]
c = 1.5e-2    # 粘性摩擦係数
J = 1.0e-3    # 慣性モーメント
```

制御目標：ステップ目標値への追従

整定時間を0.5秒以下，オーバーシュートを10%以下にし，ステップ目標値＆外乱に対して，定常偏差が生じないPID制御系を構築しなさい

制御系設計の例題

制御器として，I-PD制御を考える（定常偏差は 0 になる）



閉ループ系 $G_{yr}(s) = \dfrac{k_I}{Js^3 + (c + k_D)s^2 + (mg\ell + k_P)s + k_I}$

を**３次の規範モデル**に一致させるゲインをもとめる

$$\frac{\omega_n^3}{s^3 + \alpha_2\omega_n s^2 + \alpha_1\omega_n^2 s + \omega_n^3}$$

$$k_P = J\alpha_1\omega_n^2 - mg\ell$$

$$k_I = J\omega_n^3 \quad k_D = J\alpha_2\omega_n - c$$

制御系設計の例題

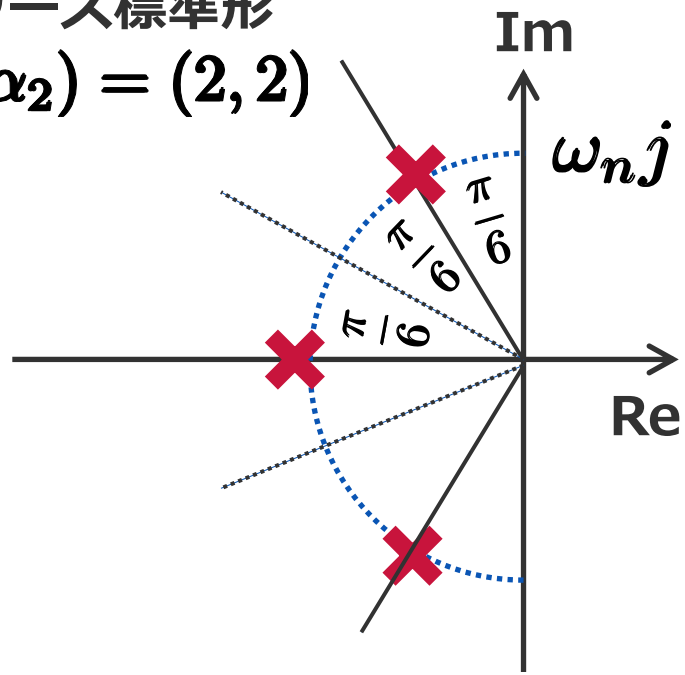
規範モデル$(\alpha_1, \alpha_2, \omega_n)$ を仕様をもとに決める

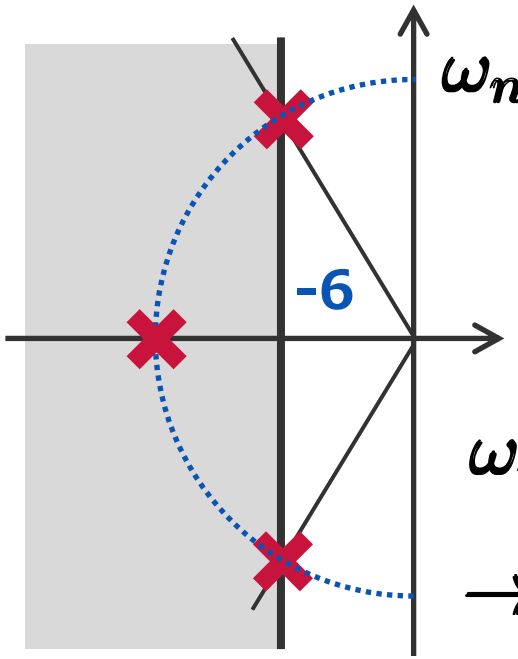
整定時間を0.5秒以下，オーバーシュートを10%以下にし，

ステップ目標値に対して，定常偏差が生じない→$|G_{yr}(0)| = 1$

バターワース標準形

$(\alpha_1, \alpha_2) = (2, 2)$





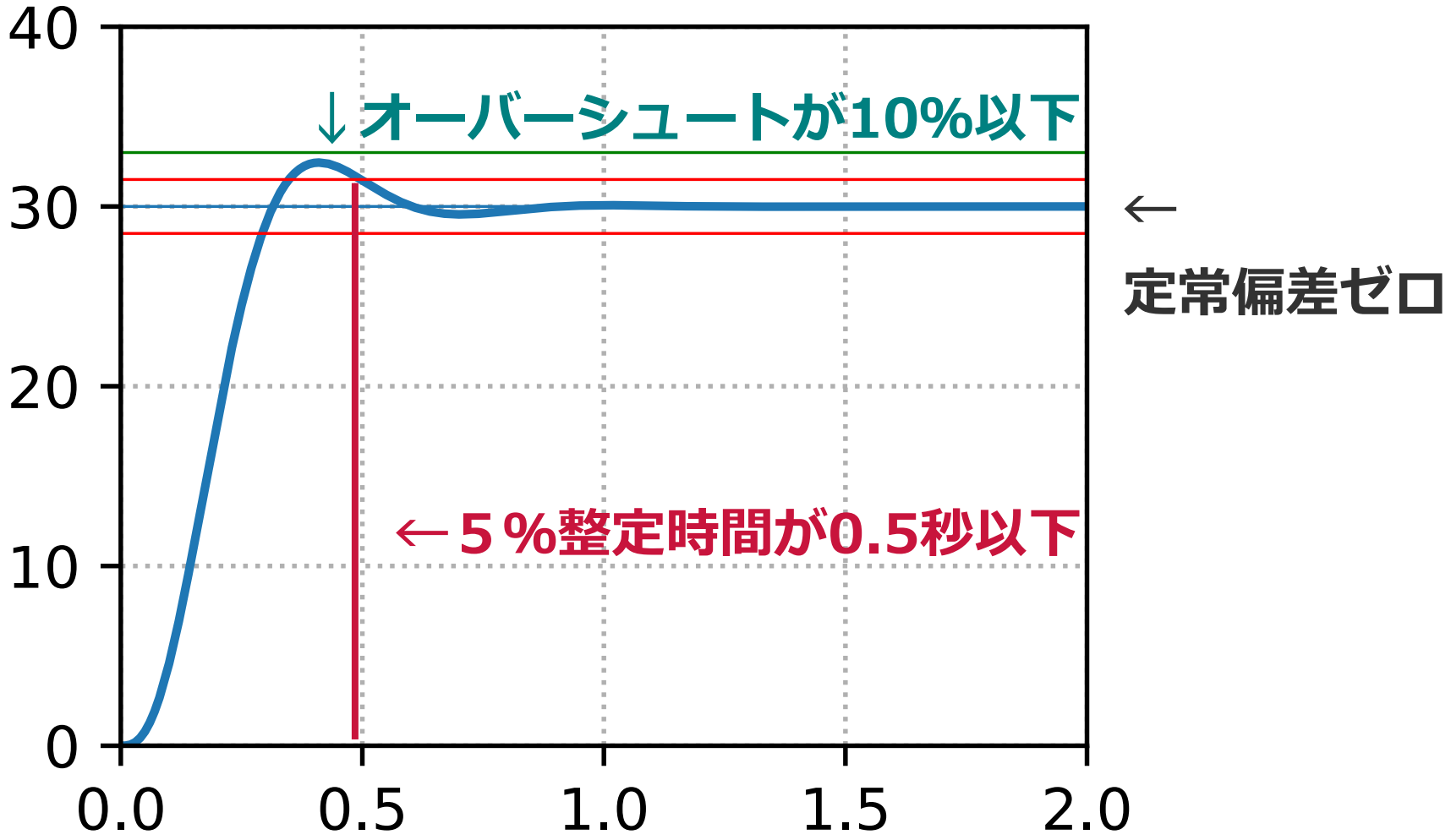
$$\omega_n \sin\frac{\pi}{6} = 6$$

$$\rightarrow \omega_n = 12$$

オーバーシュート8.2%　　実部が -3/Ts 以下で ↑ 整定時間が Ts 秒以下

制御系設計の例題

目標値を３０[deg]にしたときのステップ応答



←

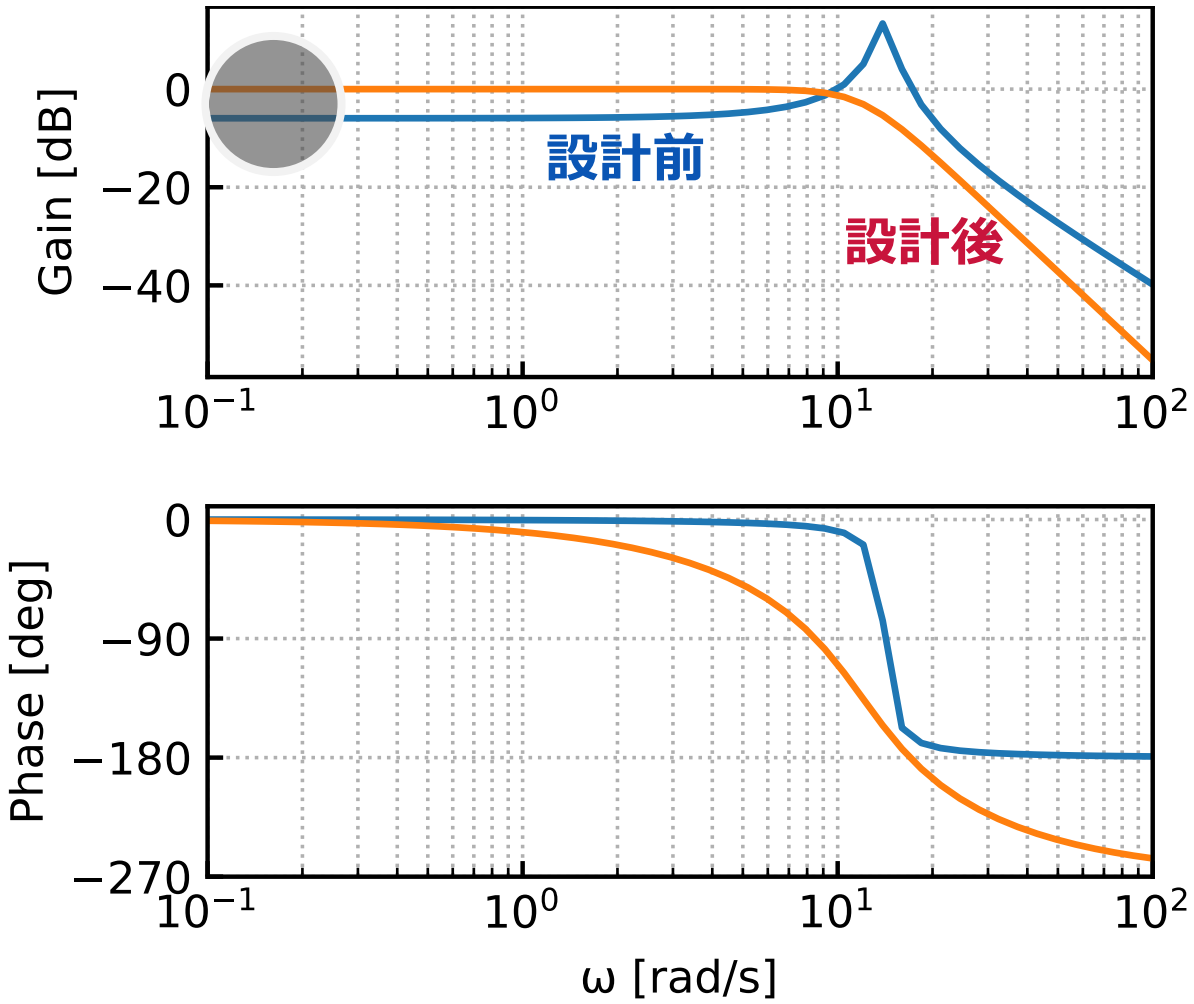
定常偏差ゼロ

制御系設計の例題

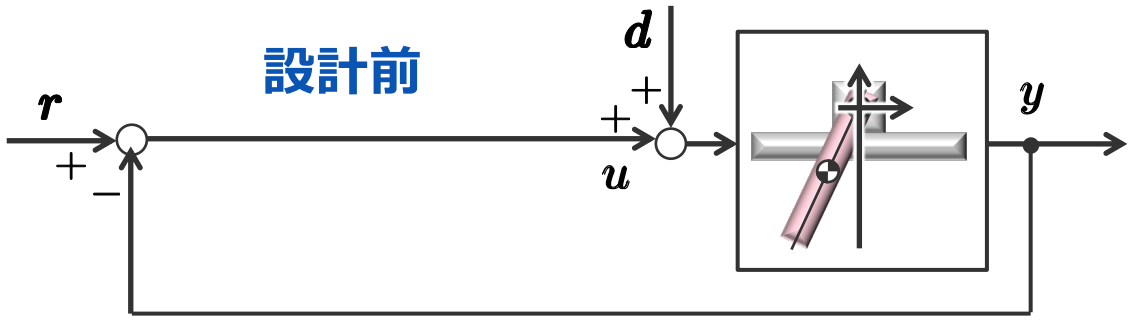
閉ループ系の周波数特性

低周波ゲインが０dB

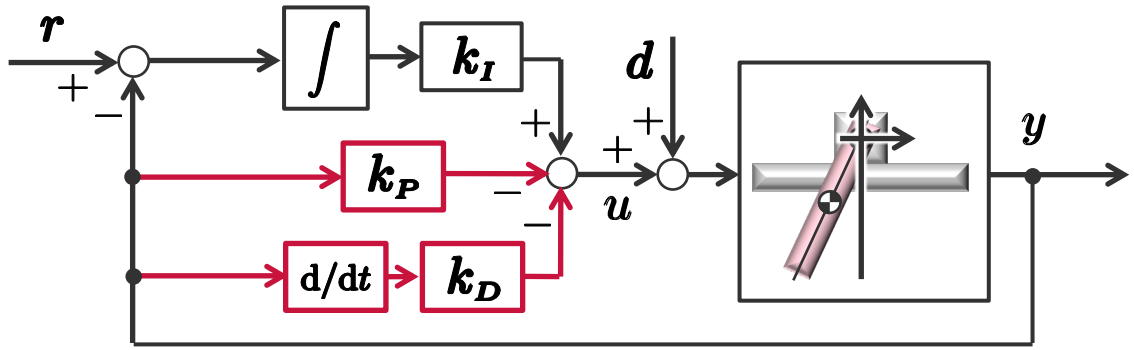
ピークゲインが小さい



設計前



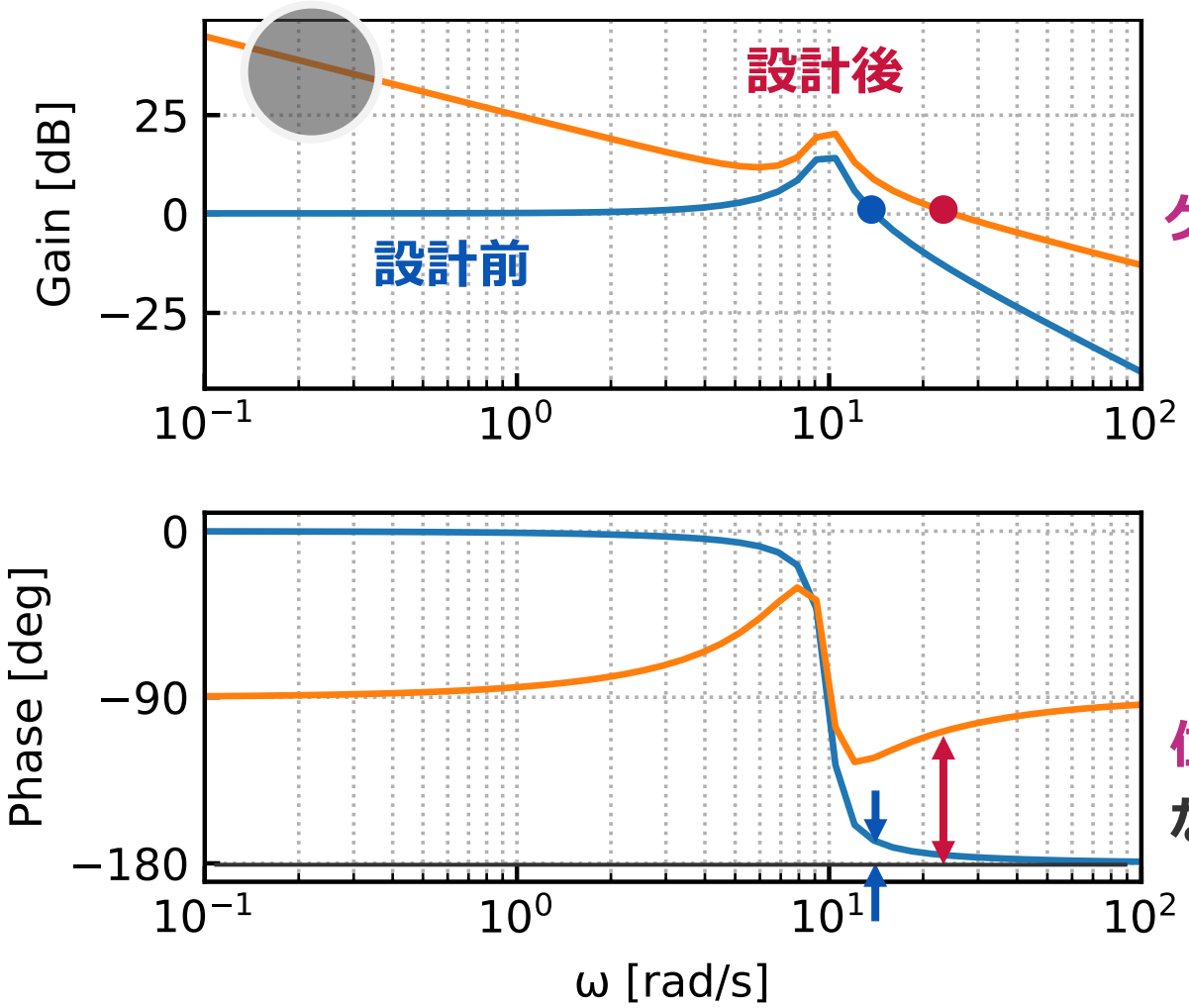
設計後



制御系設計の例題（今日の話）

開ループ系の周波数特性（ループ整形のところで登場）

↓-20dB/dec なので直流ゲインは∞になっている



ゲイン交差周波数が
大きくなっている

開ループ系の特性を見ながら
でも設計できそう！
→ ループ整形

位相余裕が大きく
なっている

勉強会の概要

Part 1

制御工学の基礎
- 制御とは，フィードバック制御，制御系設計

Pythonプログラミングの基礎
- Jupyter Notebook の使い方
- Pythonプログラミング　★ Python演習 1

制御系設計のためのモデル
- 伝達関数モデル，ブロック線図，時間応答，周波数応答　★ Python演習 2

Part 2

制御系設計のためのモデル（復習）
- 伝達関数モデル，時間応答，周波数応答

伝達関数モデルを用いた制御系設計
- 閉ループ系の設計仕様，PID制御，モデルマッチング　★ Python演習 3

ループ整形による制御系設計
- 開ループ系の設計仕様，位相進み・遅れ補償　★ Python演習 4

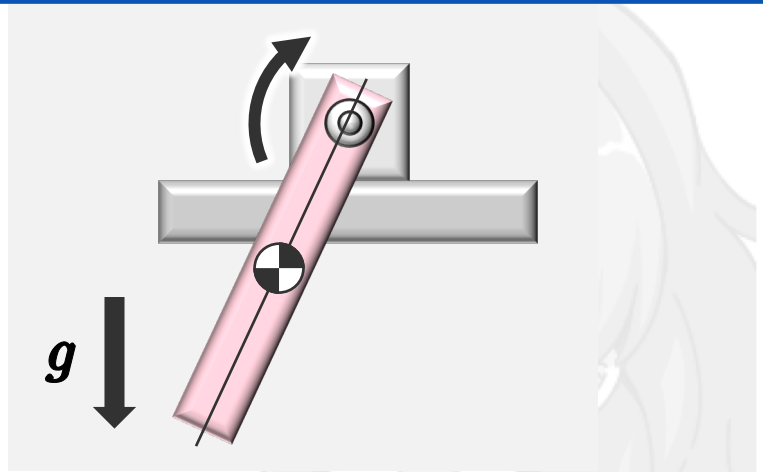
状態空間モデルを用いた制御系設計
- 状態空間モデル，状態フィードバック　★ Python演習 5

姉妹のよくある日常会話

姉：希波（きなみ）　２３歳　　妹：望結（みゆう）　１８歳

姉妹のよくある日常会話

PID制御系を設計できるように
なった気がする！



開ループ？？？ループ整形？？？？

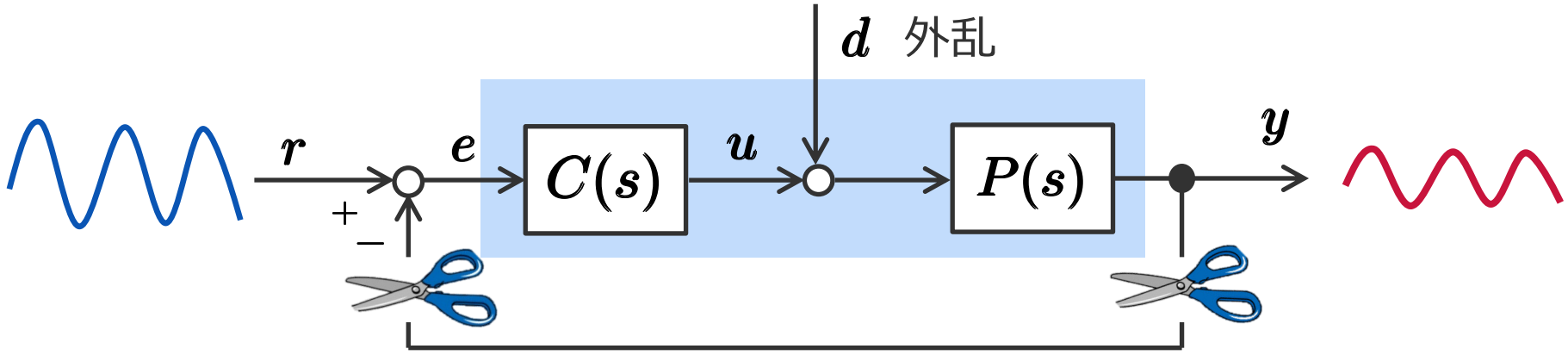
それはよかった.
でも，あとでゆっくり復習してね

つぎは，開ループ系に注目して設計
してみようか．ループ整形ってやつ

難しいかもしれないけど，開ループの
仕様とループ整形について説明するね

★ループを切って設計するために開ループ仕様を考える

ループ整形



ループを切ったときの周波数特性（開ループ系の特性）を見る

→ 設計仕様間のトレード・オフを考察しやすい

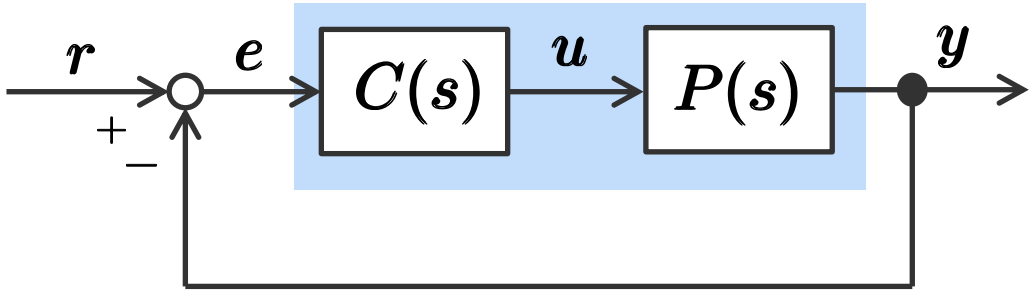
＋ 制御対象のパラメータ変動も考慮しやすくなる

モデルが未知でもボード線図が測定できさえすればよい

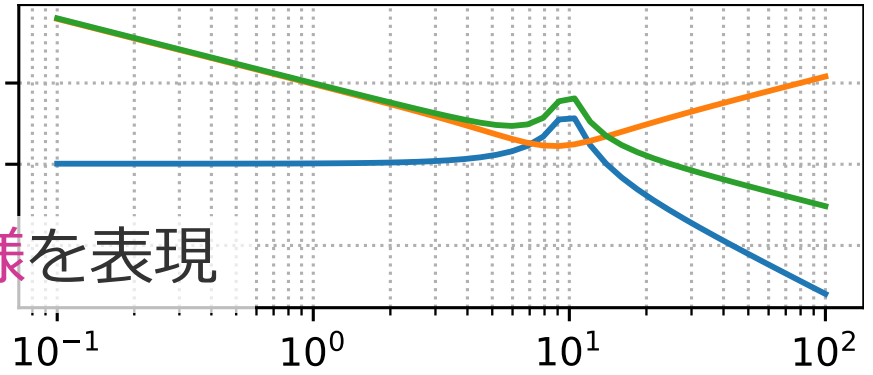
→ 制御対象の複雑さに影響されにくい

開ループ系の設計仕様

閉ループ系の設計仕様を開ループ系の仕様に書き換える



<P> + <C> ⇒ <L>



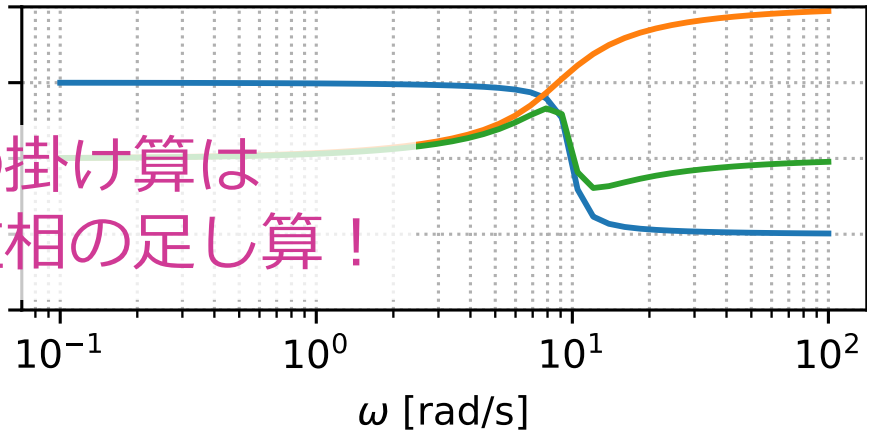
開ループ伝達関数 $L(s) = P(s)C(s)$ の性質で設計仕様を表現

（一巡伝達関数）

なぜ？ $$G_{yr}(s) = \frac{P(s)C(s)}{1 + P(s)C(s)}$$

周波数特性

→伝達関数の掛け算は
ゲインや位相の足し算！



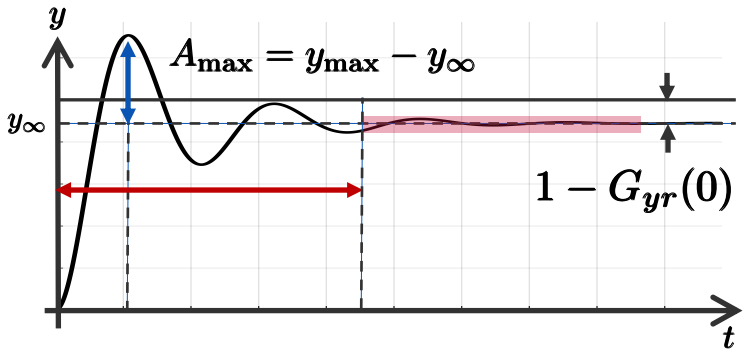
制御対象や制御器に関して非線形

- 不確かさに対する考察が容易でない
- 設計パラメータのチューニングが容易でない

閉ループ系の設計仕様（再掲）

閉ループ系の設計仕様

1）安定性 ➡ 内部安定

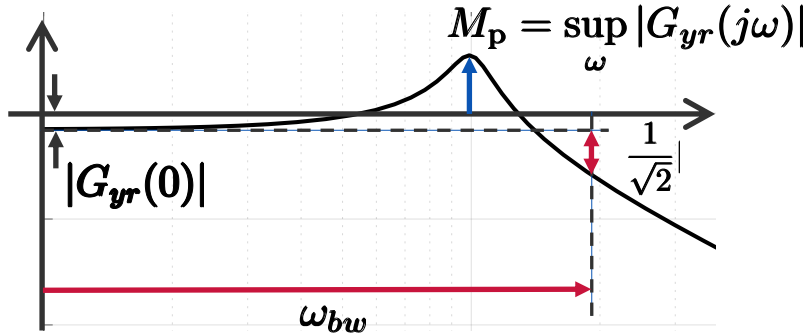


2）速応性，減衰性（≒ロバスト安定性）

- ➡ G_{yr} のステップ応答において整定時間が小さい
- ➡ 行き過ぎ量（オーバーシュート）が小さい
- ➡ G_{yr} のゲイン線図：バンド幅が十分大きい
- ➡ ピークゲイン $M_{\mathrm{p}} = \sup_{\omega} |G_{yr}(j\omega)|$ が小さい

3）定常偏差

- ➡ $|G_{yr}(0)| = 1$



開ループ系の設計仕様

開ループ系の設計仕様

1）安定性 ➡ 位相交差周波数 ＞ ゲイン交差周波数

ナイキストの安定判別法
（$L = PC$ が安定）

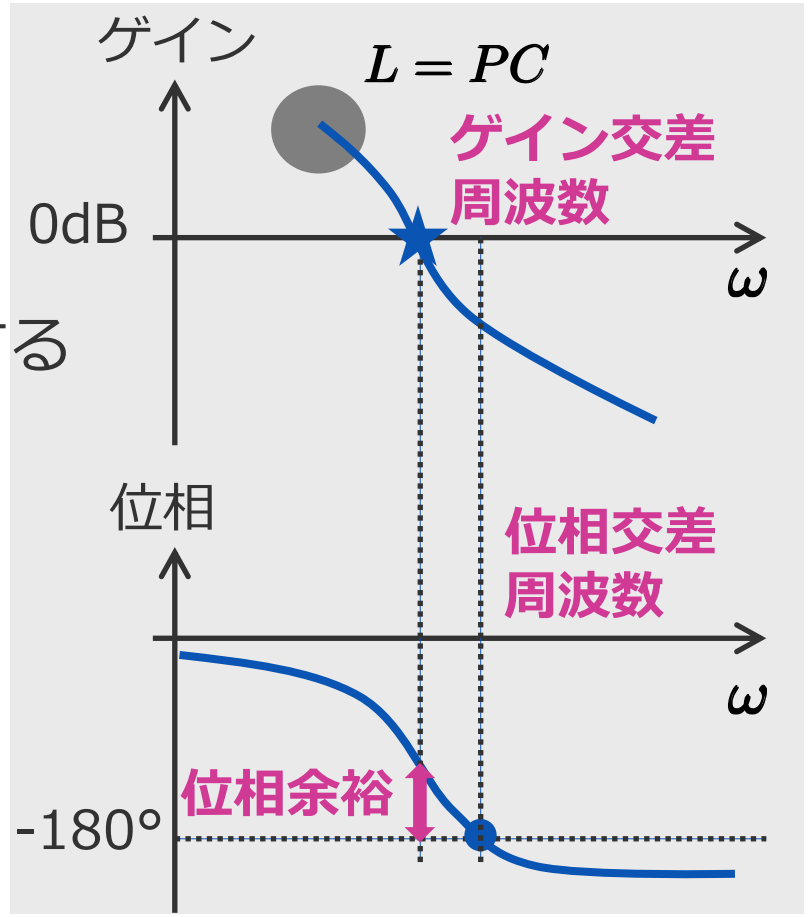
2）速応性，減衰性

⇨ ゲイン交差周波数を大きくする

➡ 位相余裕を大きくする

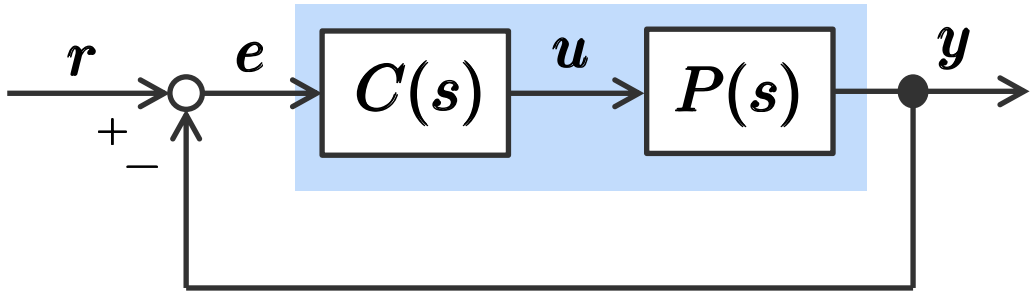
3）定常偏差

➡ $|L(0)|$ を大きくする
-20dB/dec なら１型
-40dB/dec なら２型



閉ループ系の設計仕様

1）安定性



閉ループ系が内部安定

開ループ系

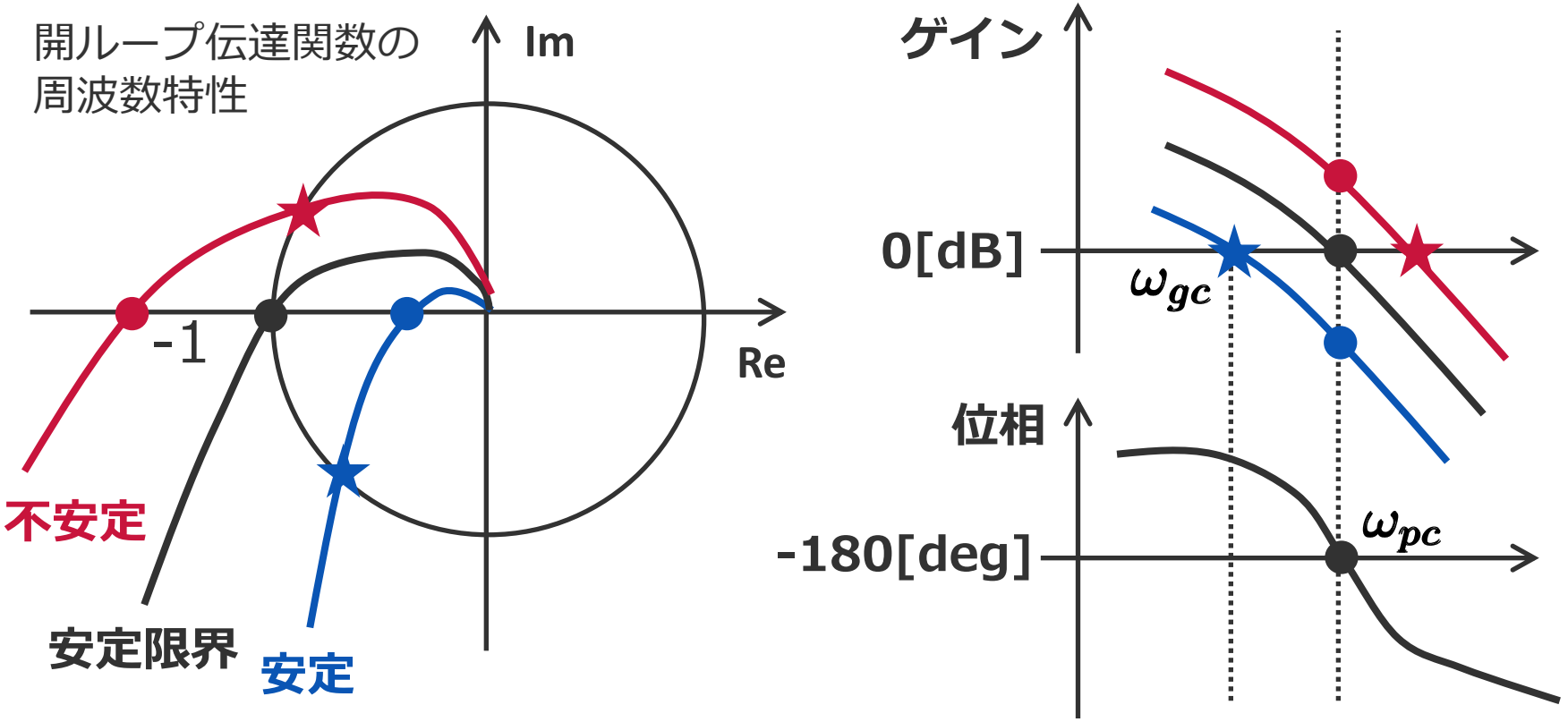
ナイキストの安定判別法（ $L = PC$ が安定）

$\omega_{pc} > \omega_{gc}$ （位相交差周波数 ＞ ゲイン交差周波数）

閉ループ系の設計仕様

1）安定性　ナイキストの安定判別法（ $L = PC$ が安定）

➡ $\omega_{pc} > \omega_{gc}$　位相交差周波数●＞ ゲイン交差周波数★



C のゲインを上げると ω_{gc} が大きくなる→安定性悪化

閉ループ系の設計仕様

2）速応性，減衰性

- G_{yr} のゲイン線図においてバンド幅が十分大きい
- ピークゲイン　$M_{\mathrm{p}} = \sup_{\omega} |G_{yr}(j\omega)|$　が小さい

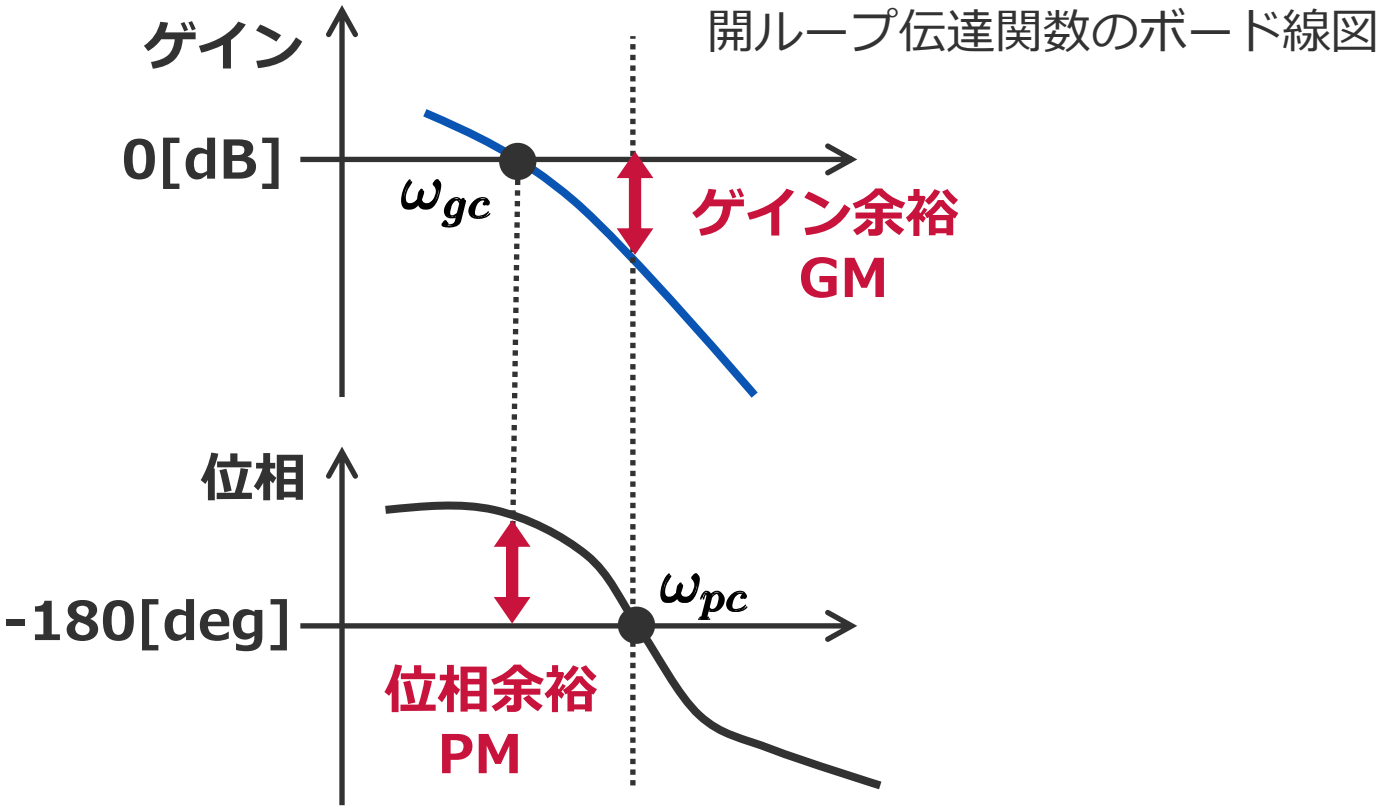
開ループ系

- 速応性

  ω_{gc} を大きくする

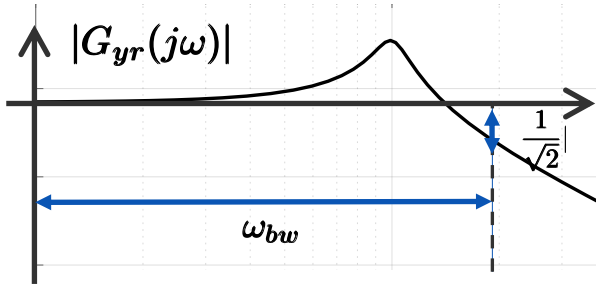
- 減衰性

  位相余裕を大きくする



閉ループ系の設計仕様

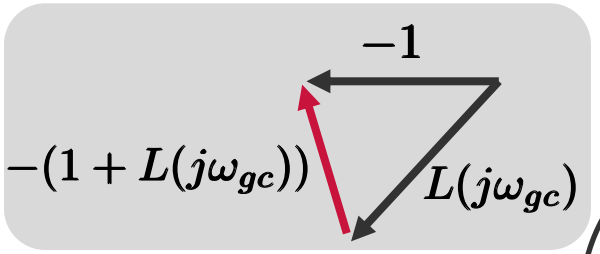
２）速応性

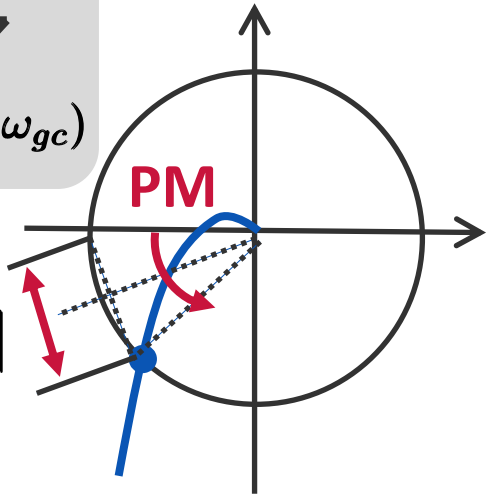


閉ループ系のゲイン $|G_{yr}(j\omega)|$
を高い周波数まで１付近にする
　＝ バンド幅をなるべく大きくしたい $|G_{yr}(j\omega_{bw})| = \frac{1}{\sqrt{2}}$　　ただし $|G_{yr}(0)| = 1$

一方，開ループ系では，$|L(j\omega_{gc})| = 1$　であり，
このとき

$$|G_{yr}(j\omega_{gc})| = \frac{1}{|1 + L(j\omega_{gc})|} = \frac{1}{2\sin\left(\frac{PM}{2}\right)}$$





PM=90　$|G_{yr}(j\omega_{gc})| = \frac{1}{\sqrt{2}}$　➡　$\omega_{gc} = \omega_{bw}$

PM<90　$|G_{yr}(j\omega_{gc})| > \frac{1}{\sqrt{2}}$　➡　$\omega_{gc} < \omega_{bw}$　(PM<90)

閉ループ系の設計仕様

2）減衰性≒ロバスト性

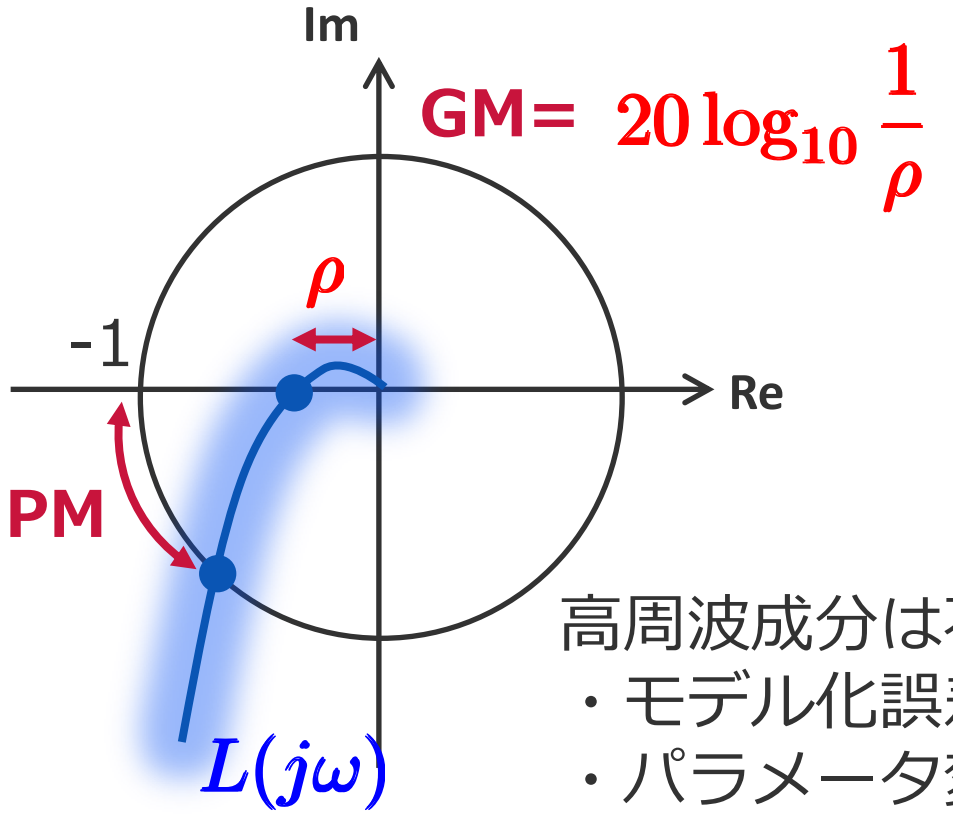
➡ ピークゲイン $M_{\mathrm{p}} = \sup_{\omega} |G_{yr}(j\omega)|$ が小さい

開ループ特性

➡ ・位相余裕 PM が大きい
・ゲイン余裕 GM が大きい

$$M_{\mathrm{p}} \geq |G_{yr}(j\omega_{gc})| = \frac{1}{2\sin\frac{PM}{2}}$$

ロバスト性が悪い
⇄ PM が小さい
⇄ $|G_{yr}(j\omega_{gc})|$ が大きい
→ ピークゲインが大きい
⇄ オーバーシュートが大きい

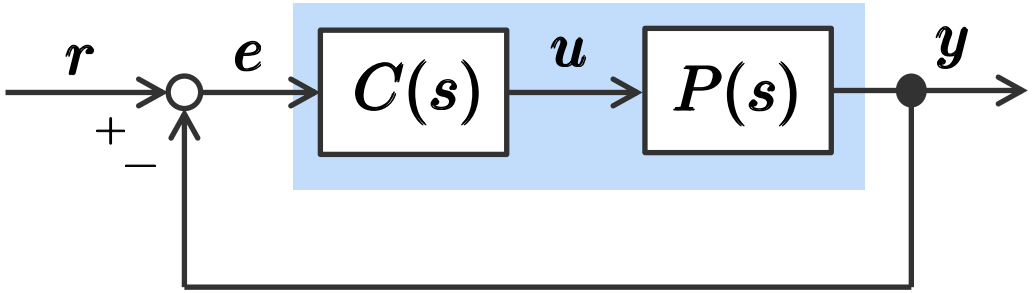


高周波成分は不確か
・モデル化誤差
・パラメータ変動

閉ループ系の設計仕様

3）定常偏差　$\omega \to 0$　のとき　$|G_{yr}(j\omega)| = 1$

➡ 低周波ゲイン $|L(0)|$ を大きくする



$$G_{er}(s) = \frac{1}{1 + P(s)C(s)} = \frac{1}{1 + L(s)}$$

$$e = G_{er} r,\ r(s) = \frac{1}{s}$$

ステップ応答の定常偏差　$e(\infty) = \lim_{s \to 0} se(s) = \lim_{s \to 0} G_{er}(s) = \dfrac{1}{1 + L(0)}$

➡ 低周波ゲイン $|L(0)|$ を大きくすれば，定常偏差が小さくなる

C に積分器が含まれていると $|L(0)| = \infty$　となり，
ステップ目標値に定常偏差なく追従する

開ループ系の設計仕様

開ループ系の設計仕様

1）安定性 ➡ 位相交差周波数 ＞ ゲイン交差周波数

ナイキストの安定判別法
（$L = PC$ が安定）

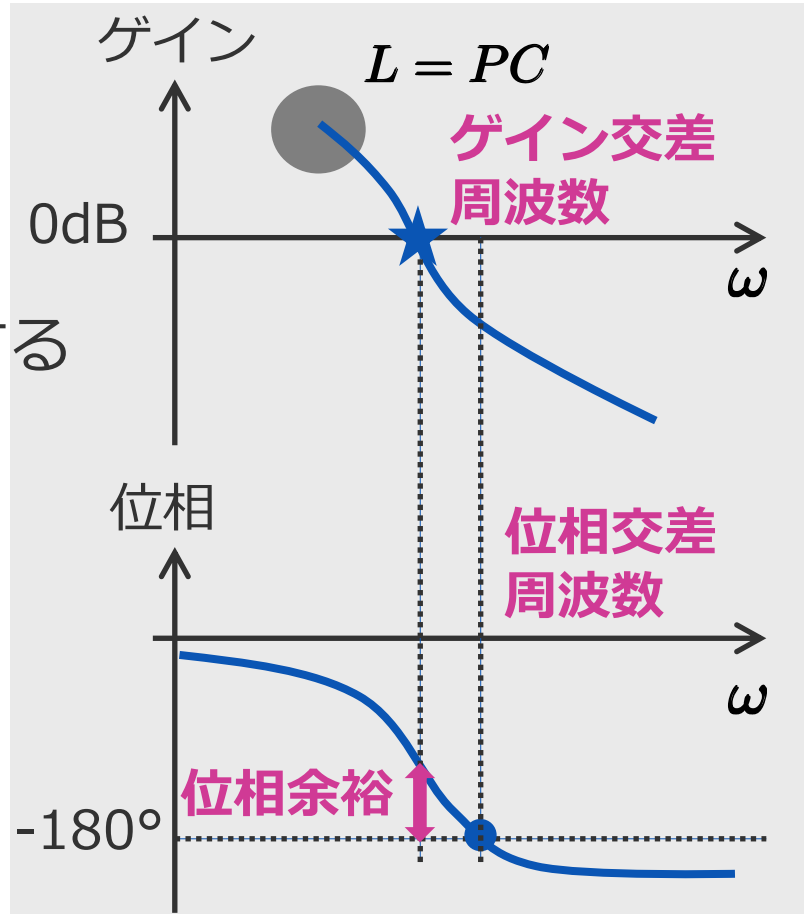
2）速応性，減衰性

➡ ゲイン交差周波数を大きくする

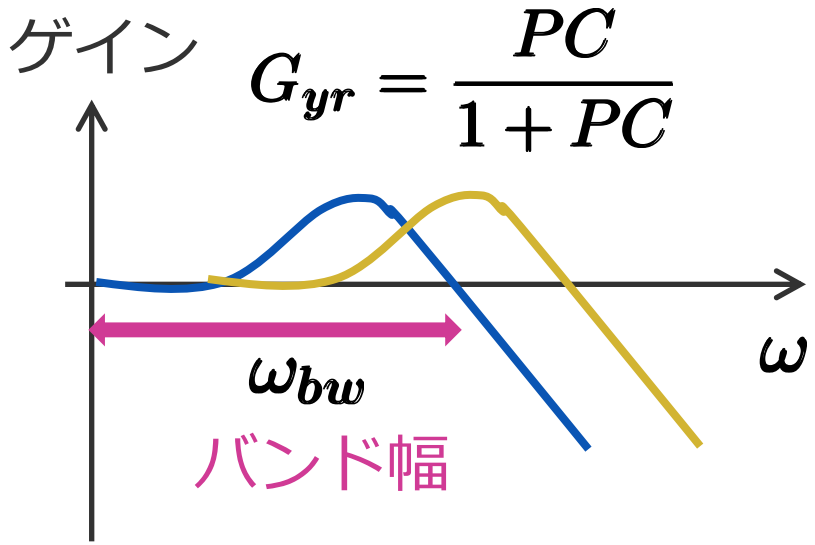
➡ 位相余裕を大きくする

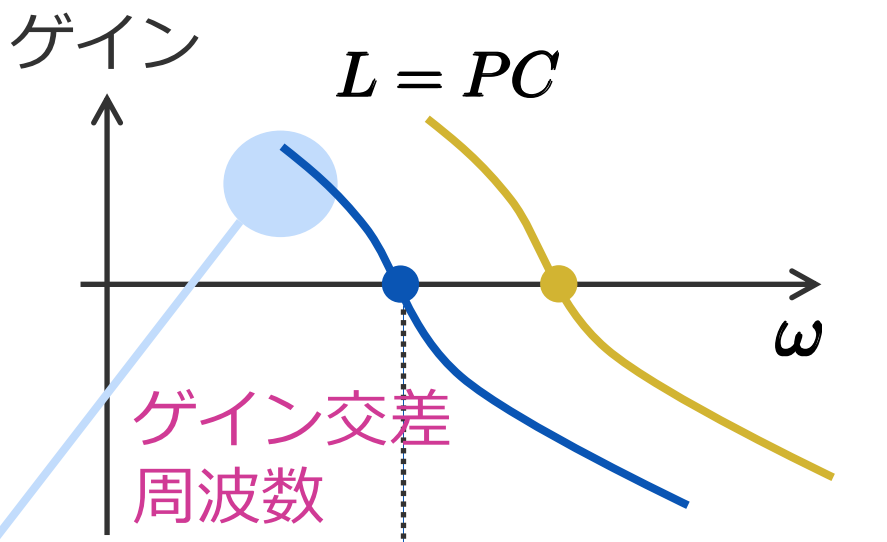
3）定常偏差

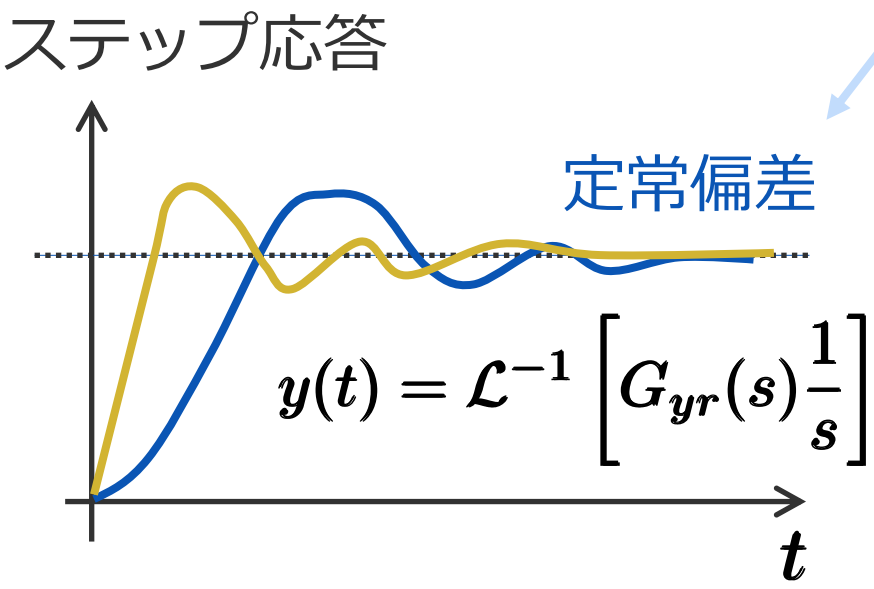
➡ $|L(0)|$ を大きくする
-20dB/dec なら１型
-40dB/dec なら２型

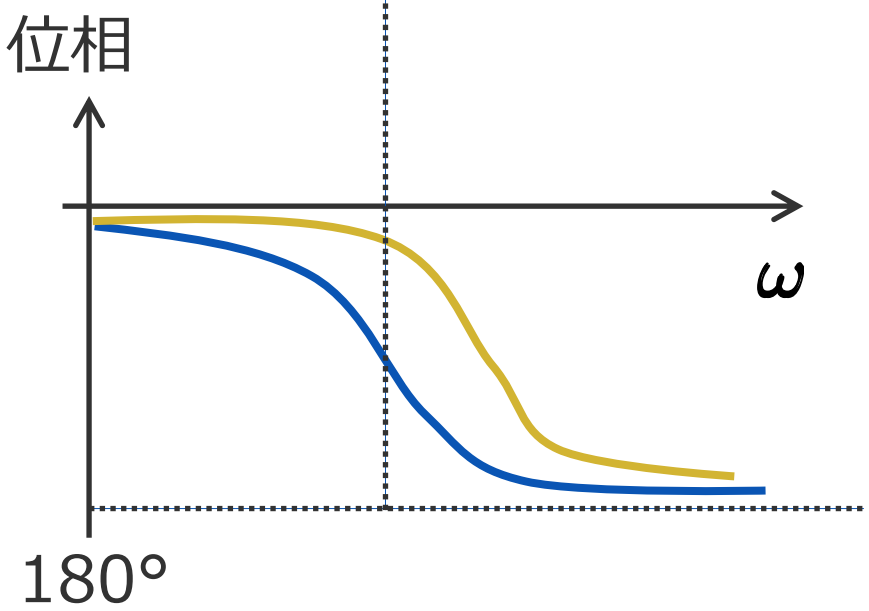


閉・開ループ系の応答特性の関係





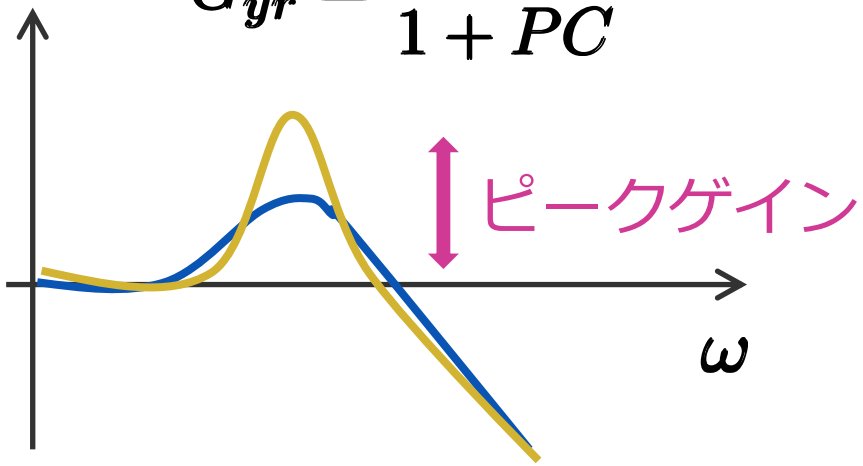




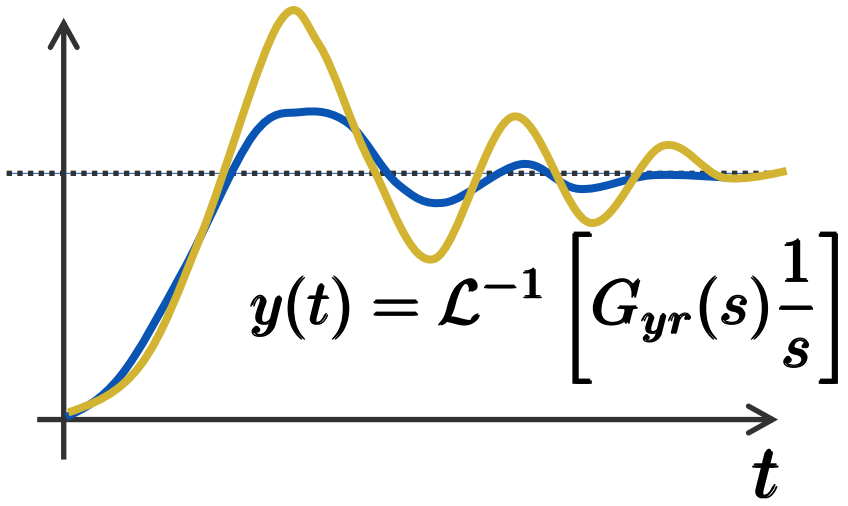
閉・開ループ系の応答特性の関係

ゲイン

$$G_{yr} = \frac{PC}{1+PC}$$

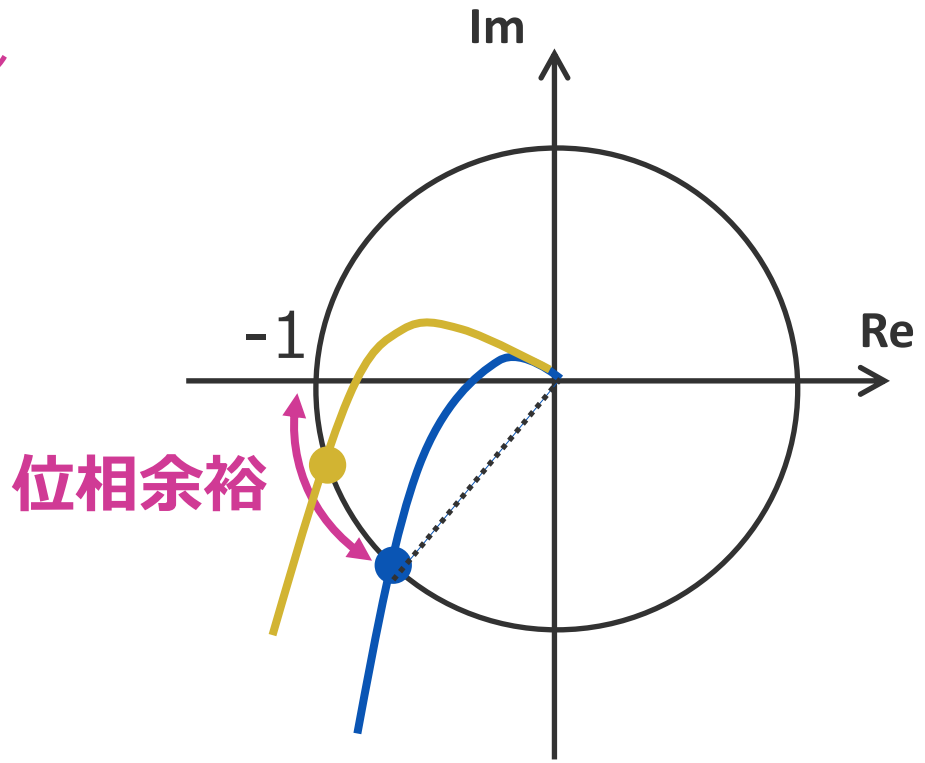


ステップ応答



ナイキスト線図

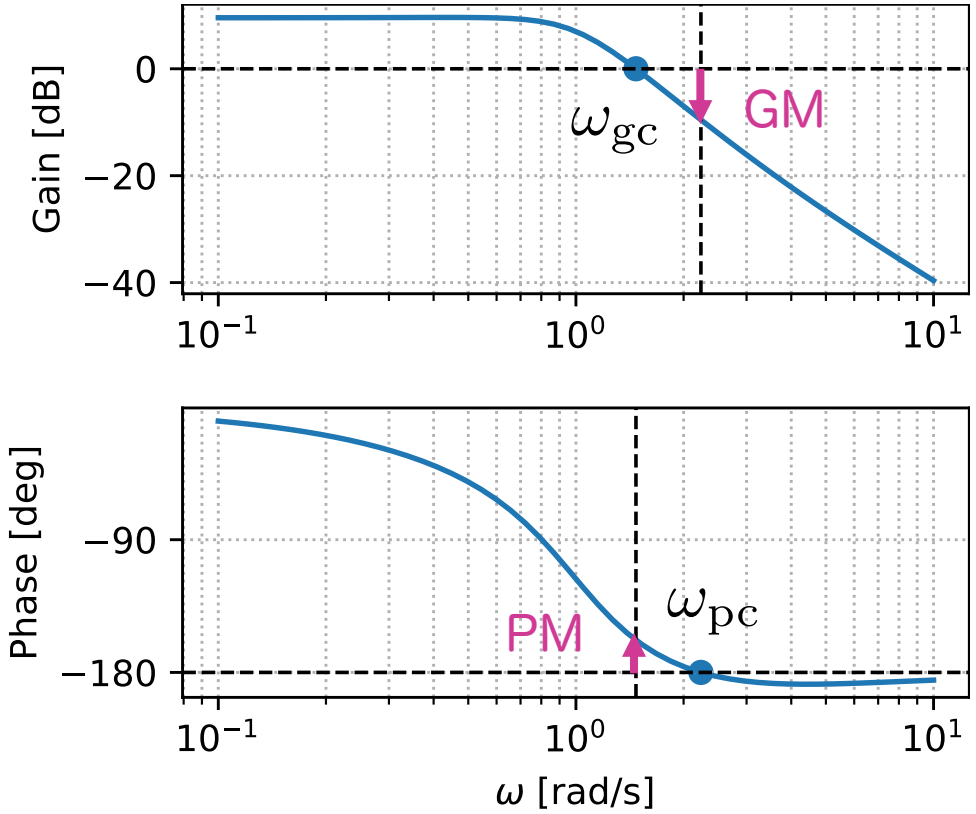
$$L = PC$$



安定余裕の計算


```
P = tf([1, 3], [1, 2, 2, 1])

fig, ax = plt.subplots(2, 1, figsize=(4, 3.5))
gain, phase, w = bode(P, logspace(-1, 1), Plot=False)
ax[0].semilogx(w, 20*np.log10(gain))
ax[1].semilogx(w, phase*180/np.pi)
```

```
GM, PM, wpc, wgc = margin(P)
print('GM=‘, 20*np.log10(GM))
print('PM=', PM)
print('wpc=', wpc)
print('wgc=', wgc)
```



GM= 9.54242509439325
PM= 21.932276766366414
wpc= 2.23606797749979
wgc= 1.4718399190801255

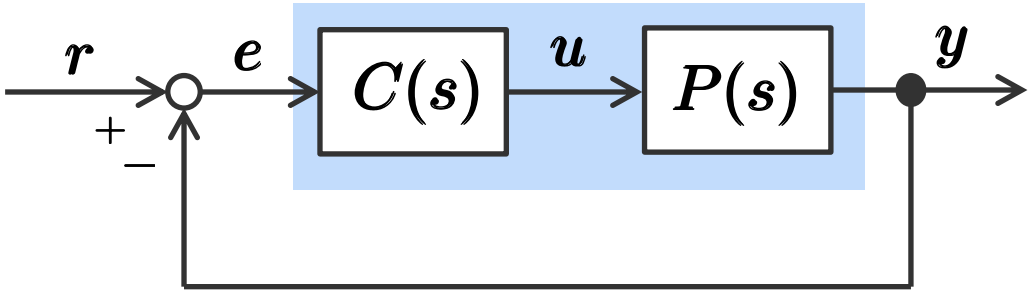
Python：安定余裕の計算

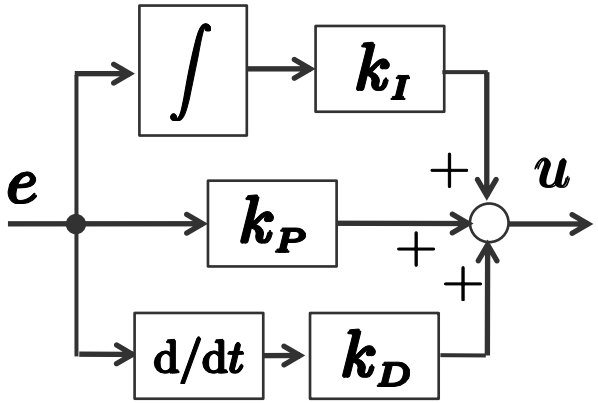
GM, PM, wpc, wgc = margin(モデル)

GM:ゲイン余裕[倍]，PM:位相余裕[deg]，
wpc: 位相交差周波数[rad/s]，wgc: ゲイン交差周波数

ループ整形法

PID補償





$$u(t) = k_P e(t) + k_I \int_0^t e(t)\mathrm{d}t + k_D \frac{\mathrm{d}}{\mathrm{d}t} e(t)$$

$$u(s) = C(s)e(s) \quad C(s) = k_P + \frac{k_I}{s} + k_D s$$

P　　I　　D

Proportional　　P制御：速応性に寄与

Integral　　I制御：定常偏差改善，位相が遅れる

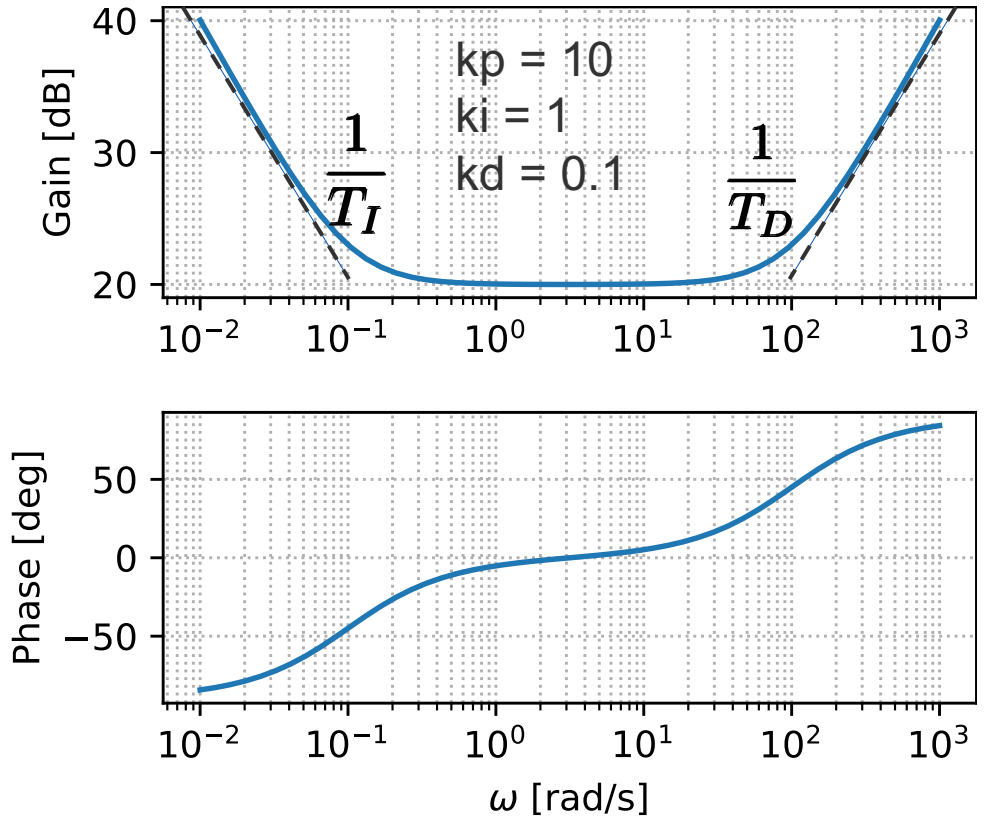
Derivative　　D制御：位相が進む，ノイズに敏感

PID制御器の周波数特性

並列型 $$C(s) = k_P + \frac{k_I}{s} + k_D s$$

$$= k_P \left(1 + \frac{1}{T_I s} + T_D s\right)$$

$$\frac{1}{T_I} = \frac{k_I}{k_P} \qquad \frac{1}{T_D} = \frac{k_P}{k_D}$$



P制御により

$20 \log_{10} k_P$ だけゲインが大きなる

I制御により，$1/T_I$ [rad/s] 以下の

・ゲインが大きなる

・位相が遅れる（振動的になりやすい）

D制御により，$1/T_D$ [rad/s] 以上の

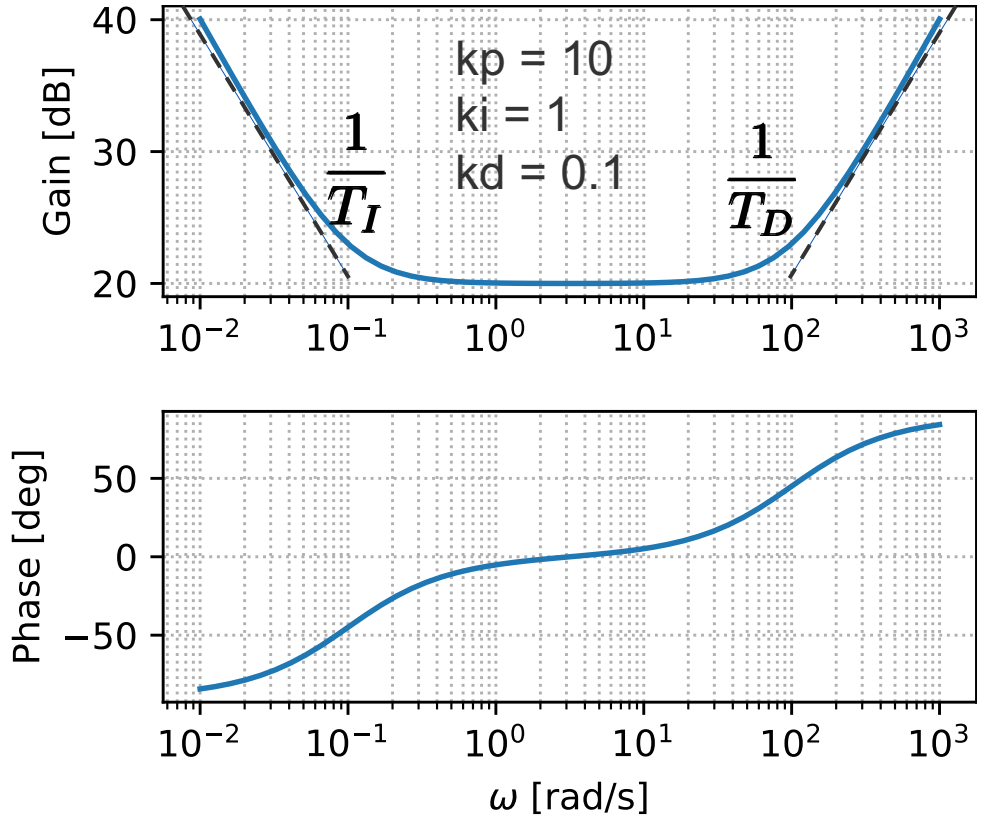
・ゲインが大きなる（ノイズの影響を受ける）

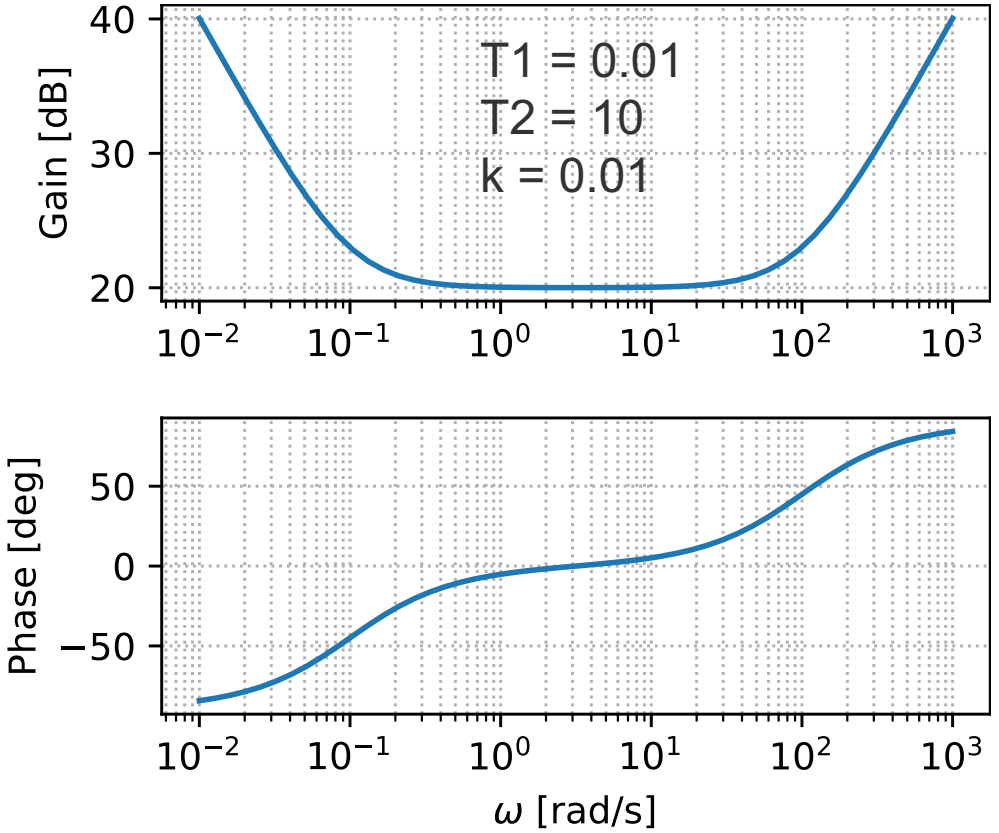
・位相が進む

PID制御器の周波数特性

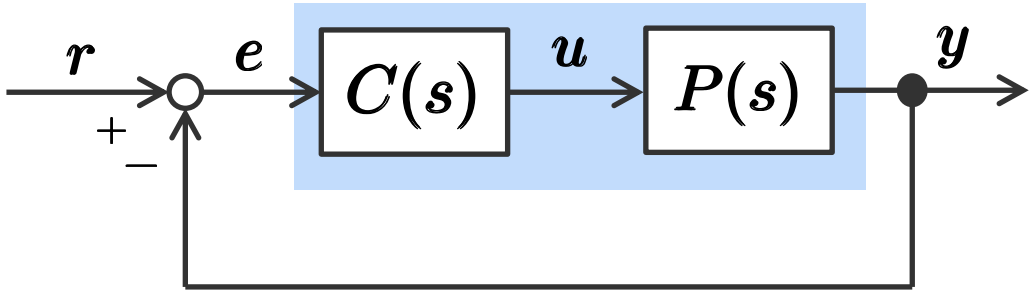
並列型 $$C(s) = k_P + \frac{k_I}{s} + k_D s = k_P \left(1 + \frac{1}{T_I s} + T_D s\right)$$

直列型 $$C(s) = k\left(1 + \frac{1}{T_1 s}\right)(1 + T_2 s)$$



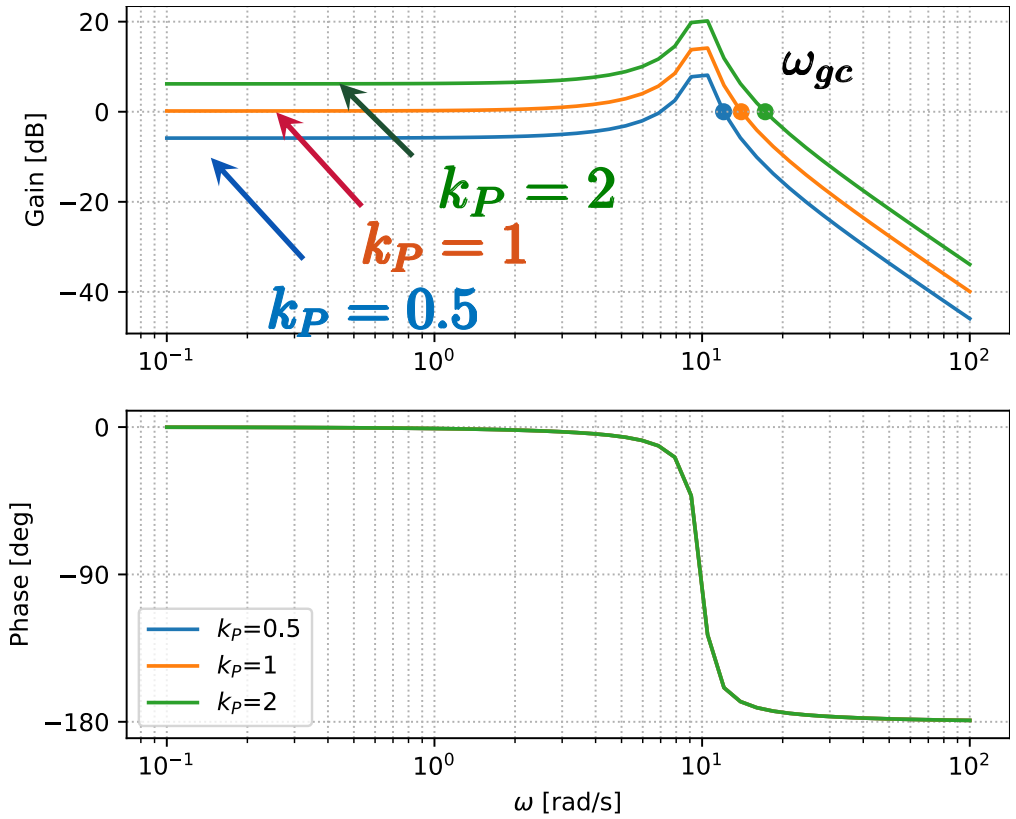


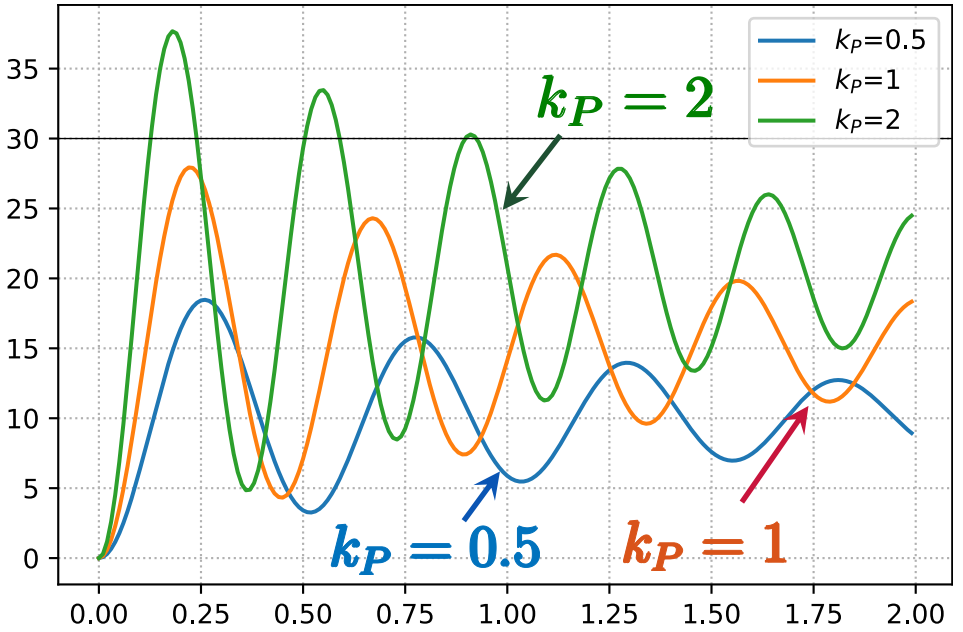
P制御



$$P(s) = \frac{1}{Js^2 + cs + mg\ell}$$

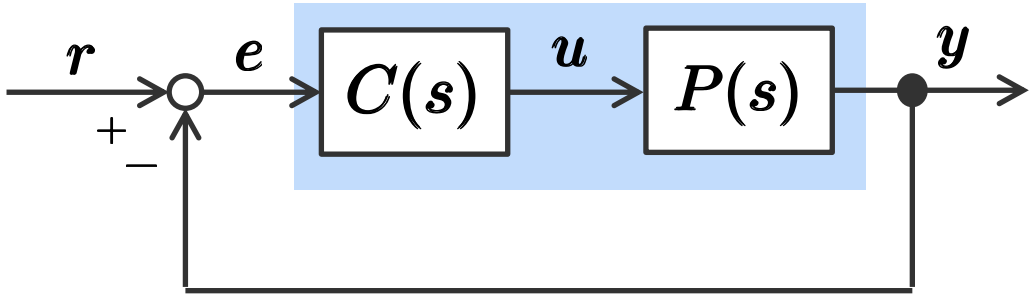
$$C(s) = k_P$$





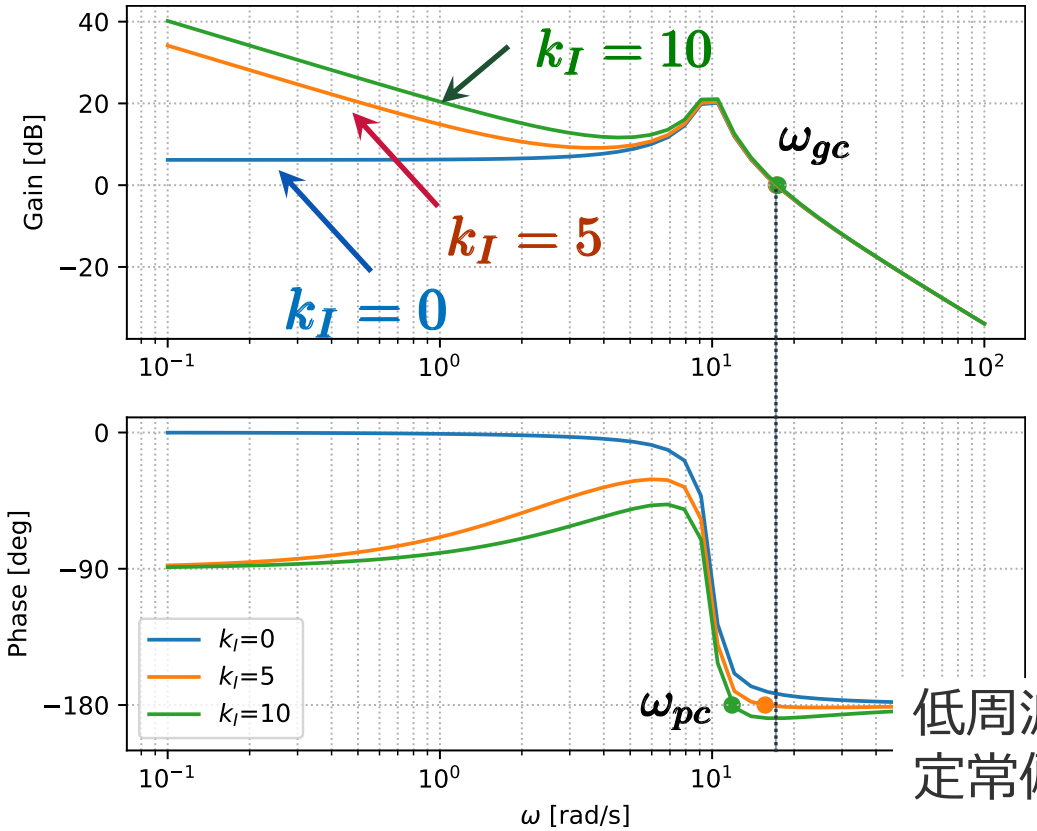
ゲイン交差が大きくなり応答が早くなるが，
位相余裕が小さくなるため，振動的になる

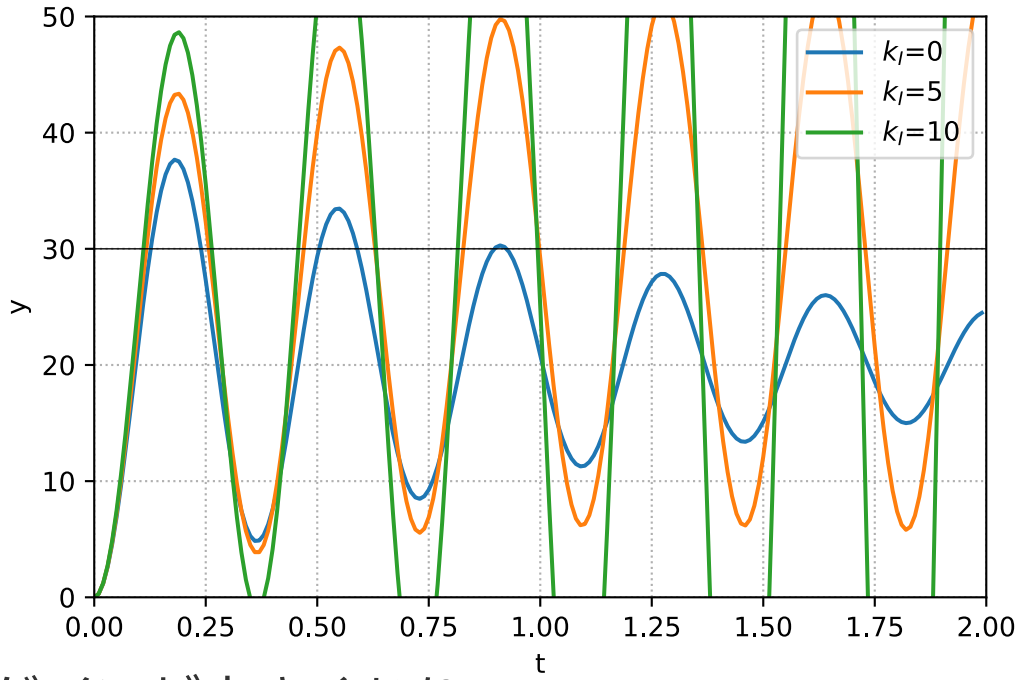
PI制御



$$P(s) = \frac{1}{Js^2 + cs + mg\ell}$$

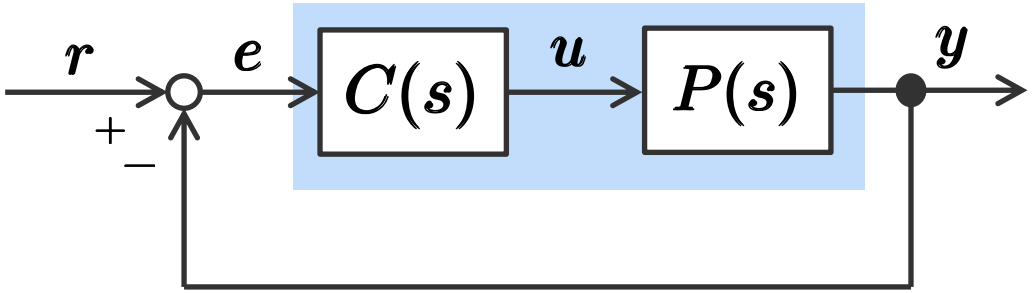
$$C(s) = k_P + \frac{k_I}{s} \qquad k_P = 2$$





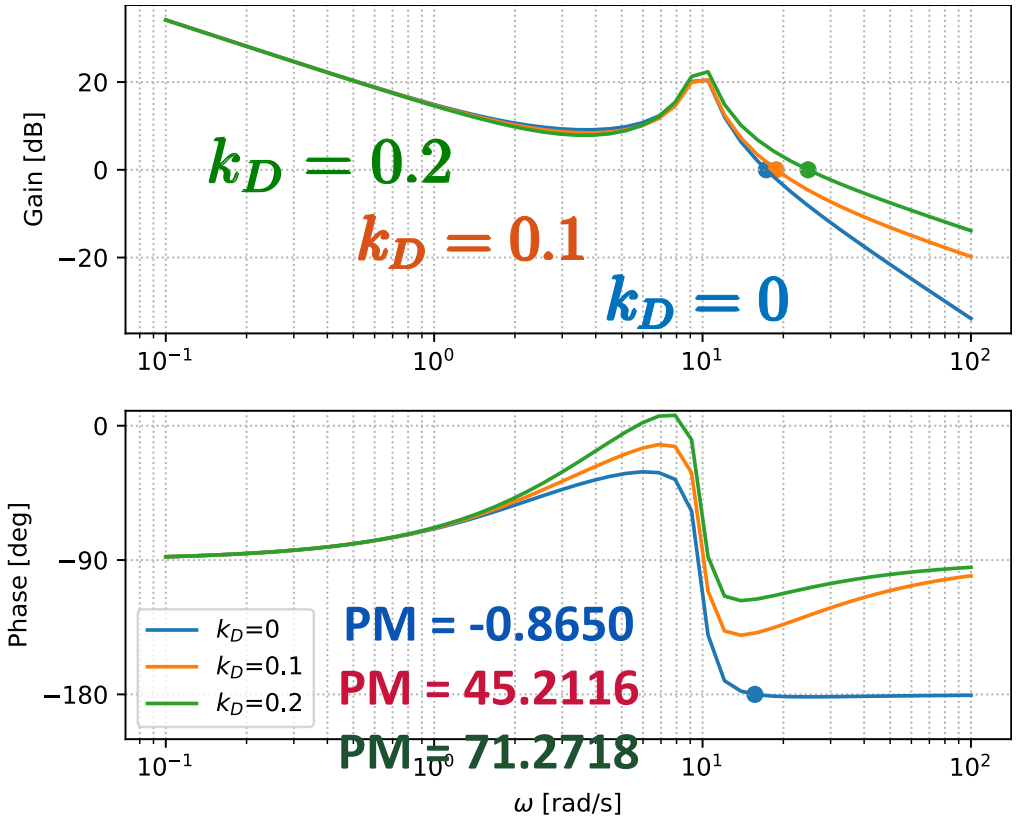
低周波ゲインが大きくなり，
定常偏差が小さくなる．その一方，振動的（or 不安定）になる．

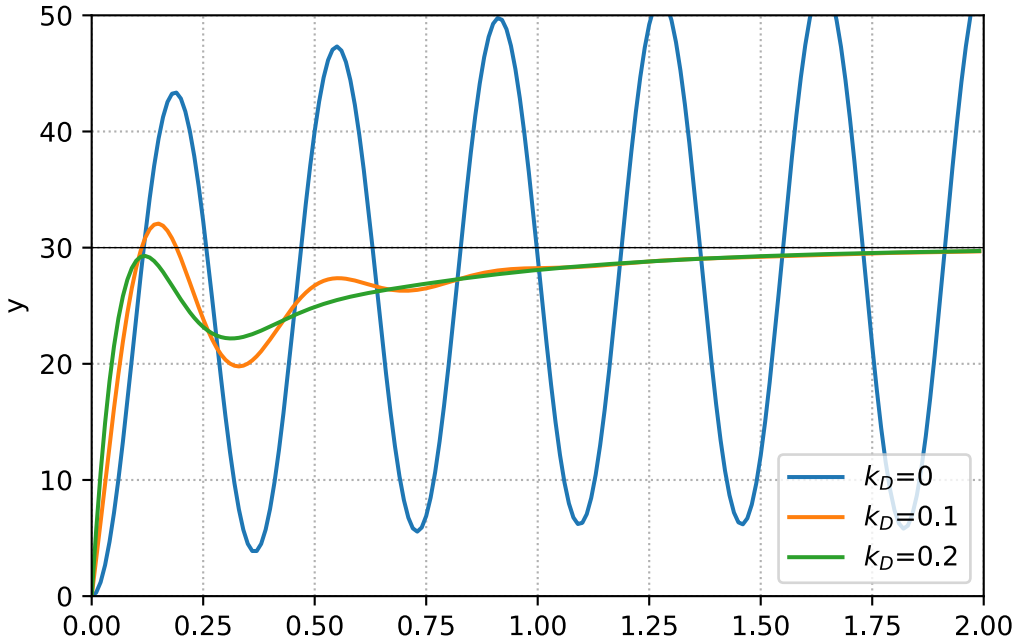
PID制御



$$P(s) = \frac{1}{Js^2 + cs + mg\ell}$$

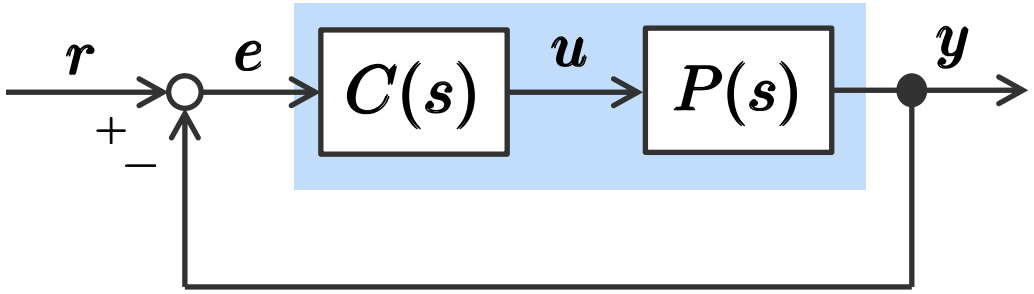
$$C(s) = k_P + \frac{k_I}{s} + k_D s \qquad k_P = 2, \quad k_I = 5$$





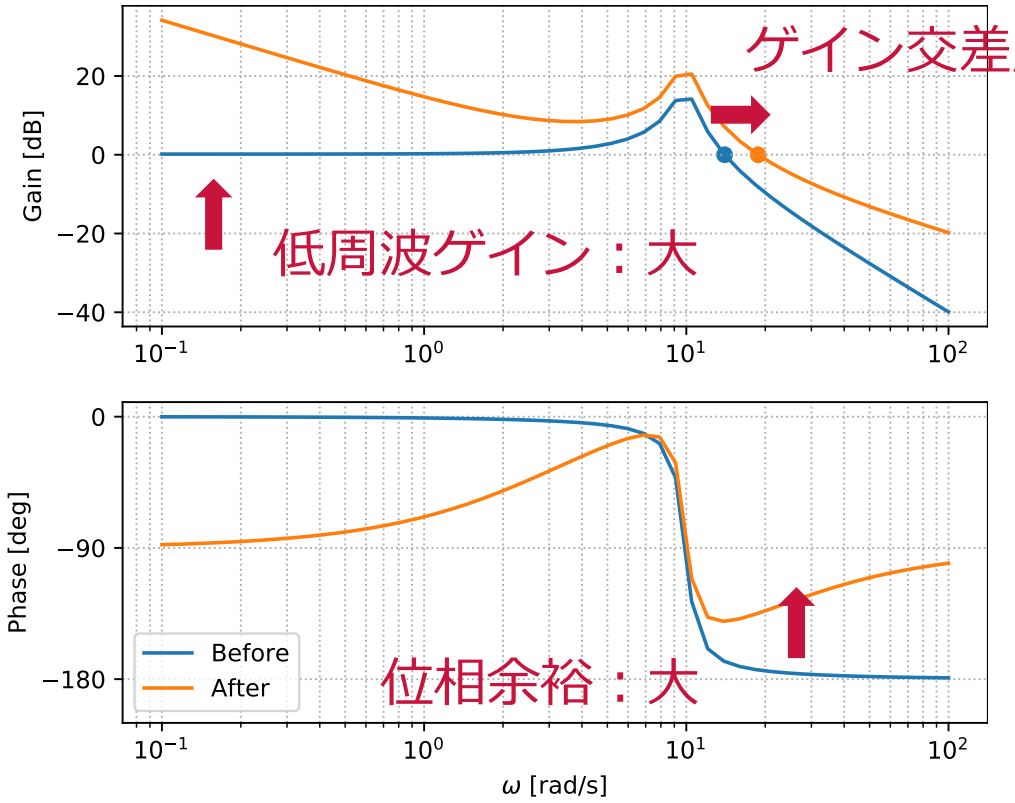
位相余裕を大きくすることで，
振動を抑えることができる

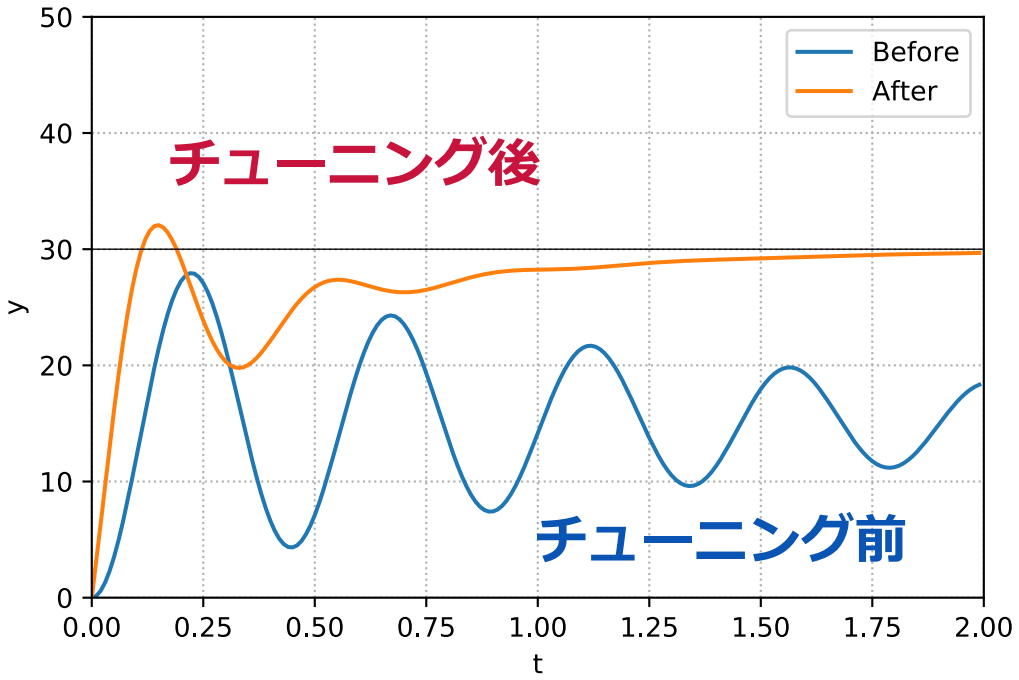
PID制御



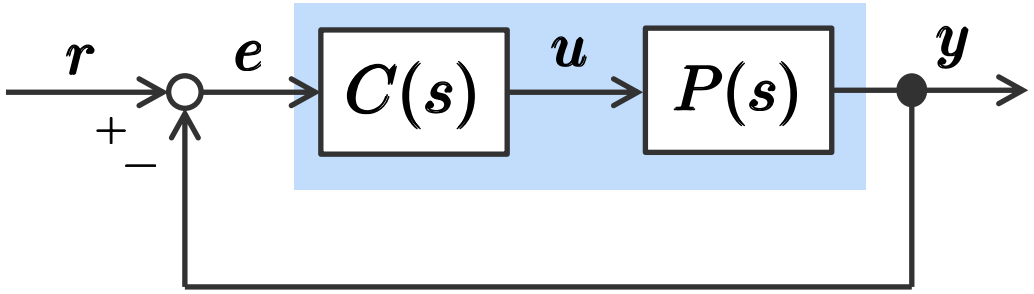
$$P(s) = \frac{1}{Js^2 + cs + mg\ell}$$

$$C(s) = k_P + \frac{k_I}{s} + k_D s$$



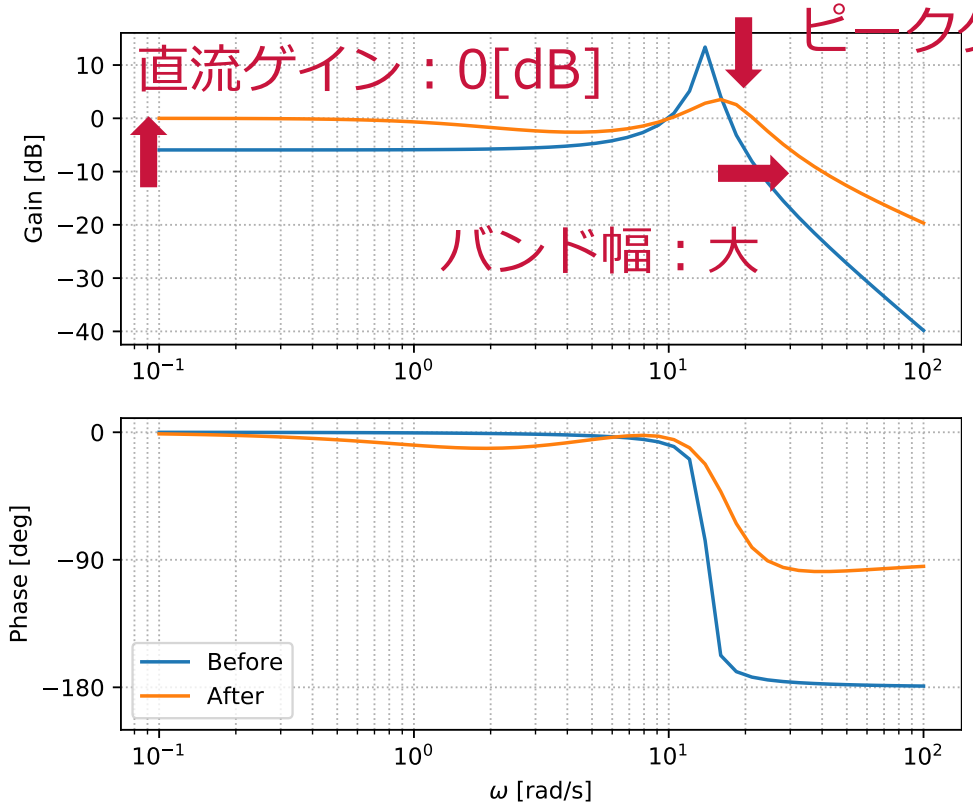


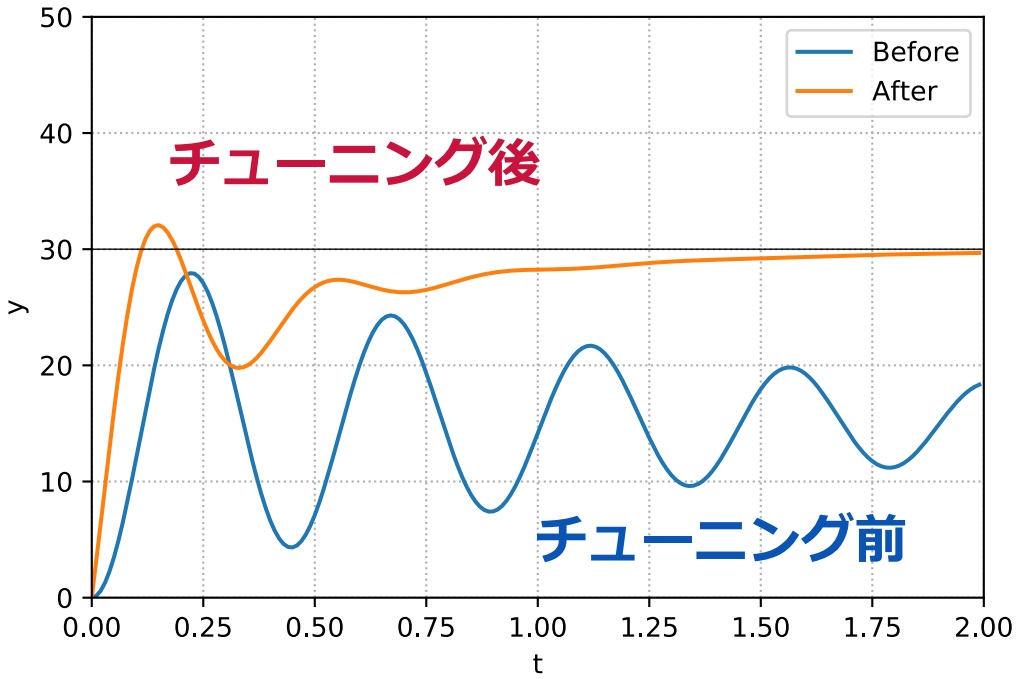
PID制御



$$P(s) = \frac{1}{Js^2 + cs + mg\ell}$$

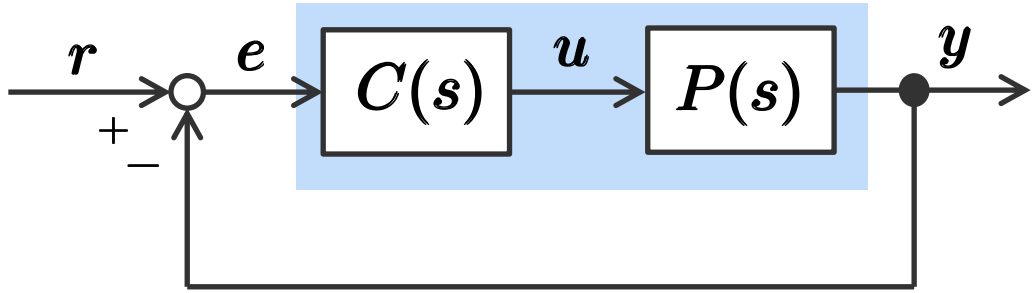
$$C(s) = k_P + \frac{k_I}{s} + k_D s$$

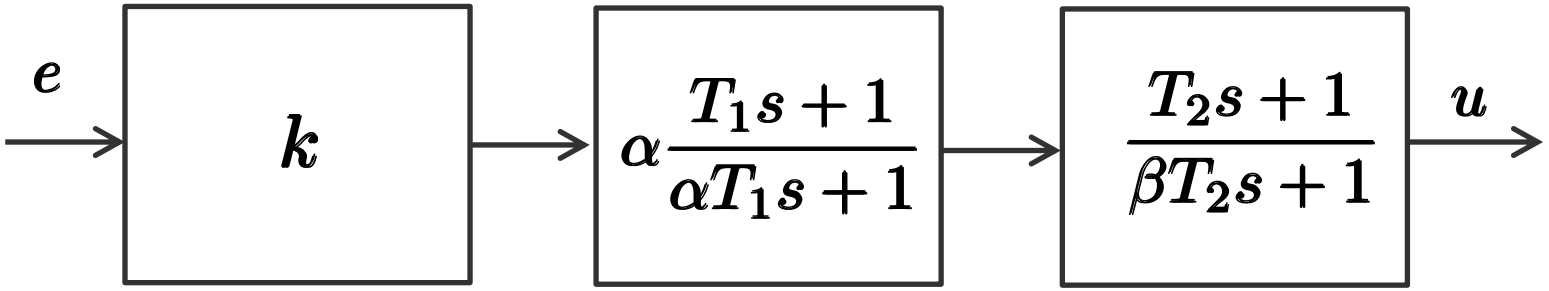




ループ整形法

直列補償によるループ整形 ⟷ PID制御は並列補償





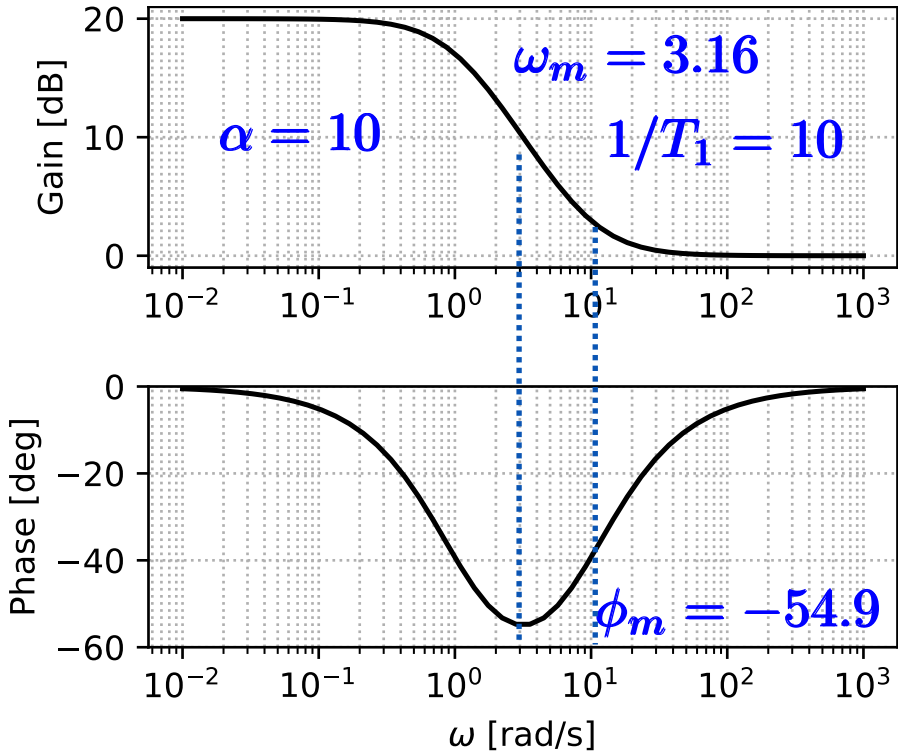
ゲイン補償　　位相遅れ補償　　位相進み補償

定常偏差低減，速応性の改善，安定性悪化

位相遅れ補償と位相進み補償

位相遅れ補償

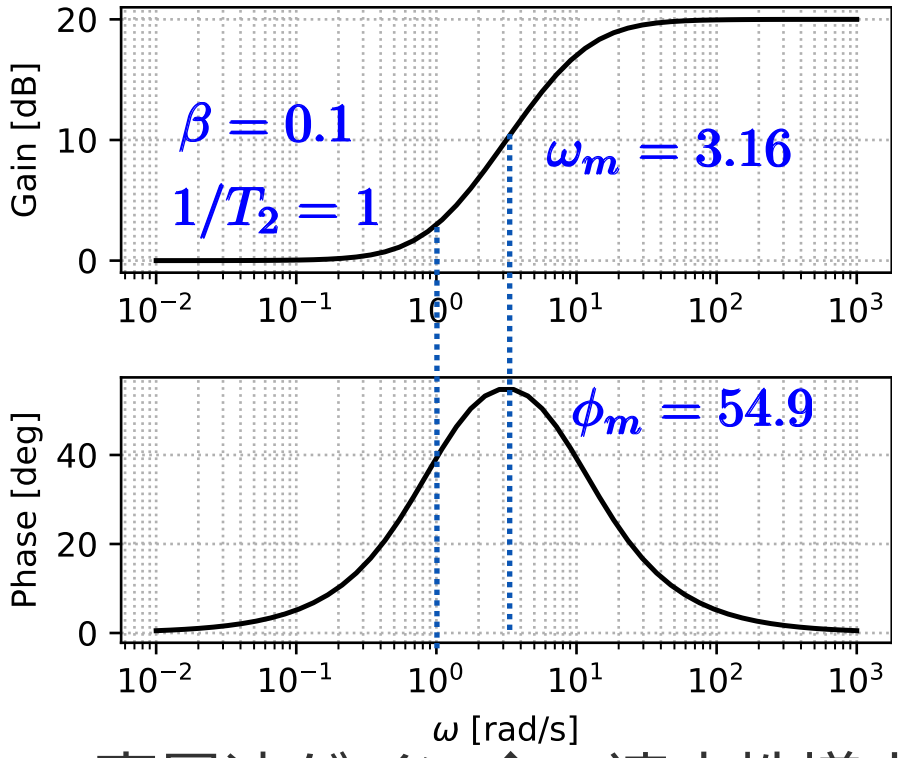
$$C_1(s) = \alpha \frac{T_1 s + 1}{\alpha T_1 s + 1} \quad \alpha > 1$$



低周波ゲイン↑：定常偏差低減
位相が遅れる（安定性悪化）

位相進み補償

$$C_2(s) = \frac{T_2 s + 1}{\beta T_2 s + 1} \quad \beta < 1$$



高周波ゲイン↑：速応性増大
位相余裕を増大（安定性改善）

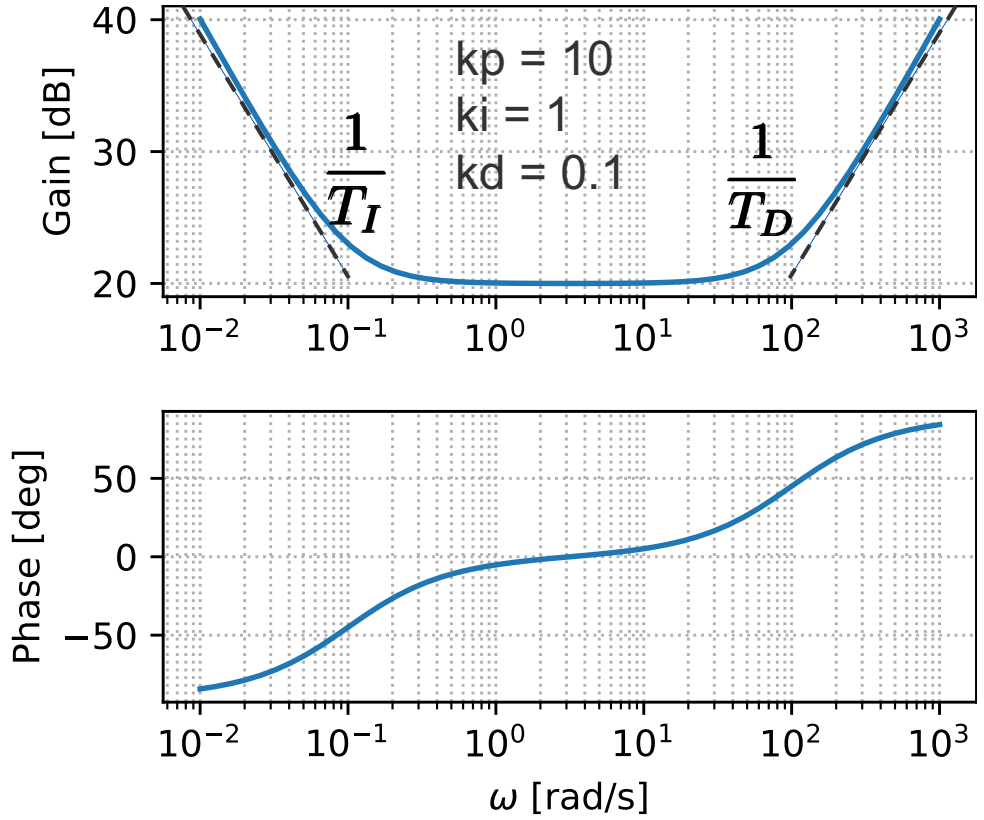
位相進み・遅れ補償の周波数特性

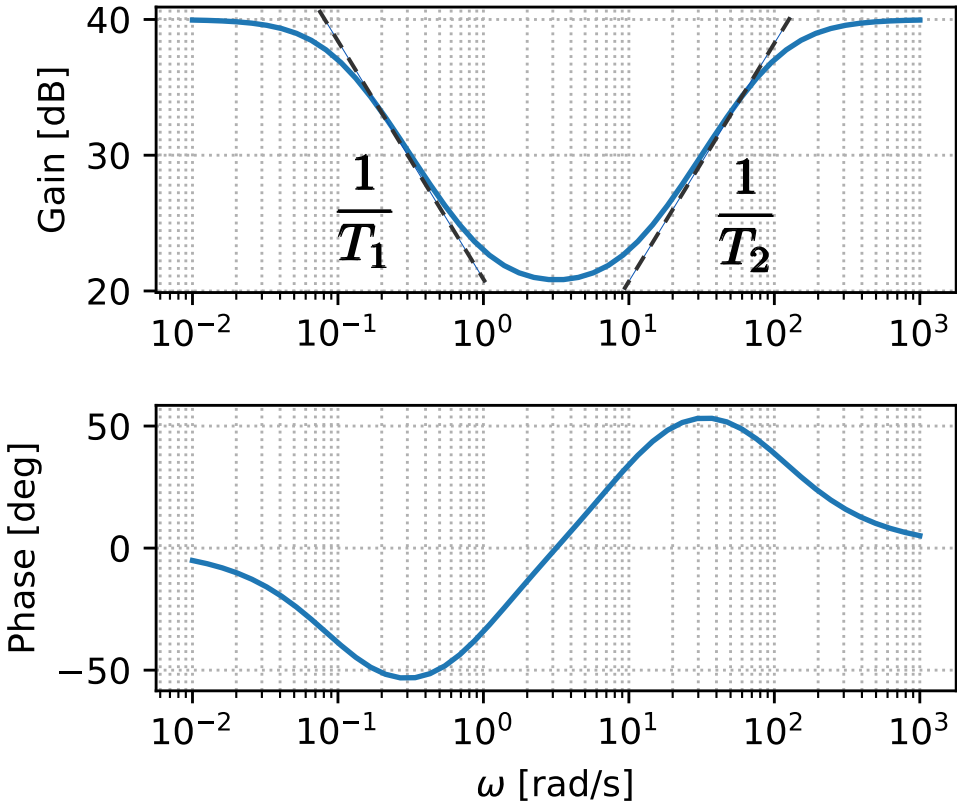
PID制御 $C(s) = k_P + \frac{k_I}{s} + k_D s$

$= k_P \left(1 + \frac{1}{T_I s} + T_D s\right)$

位相進み・位相遅れ補償

$C(s) = k\left(\alpha \frac{T_1 s + 1}{\alpha T_1 s + 1}\right)\left(\frac{T_2 s + 1}{\beta T_2 s + 1}\right)$



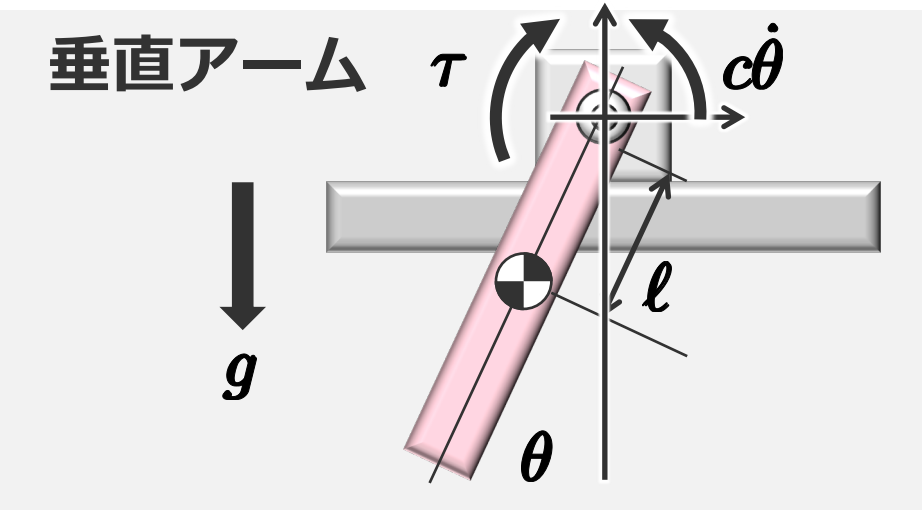


k = 10

alpha = 10
T1 = 1

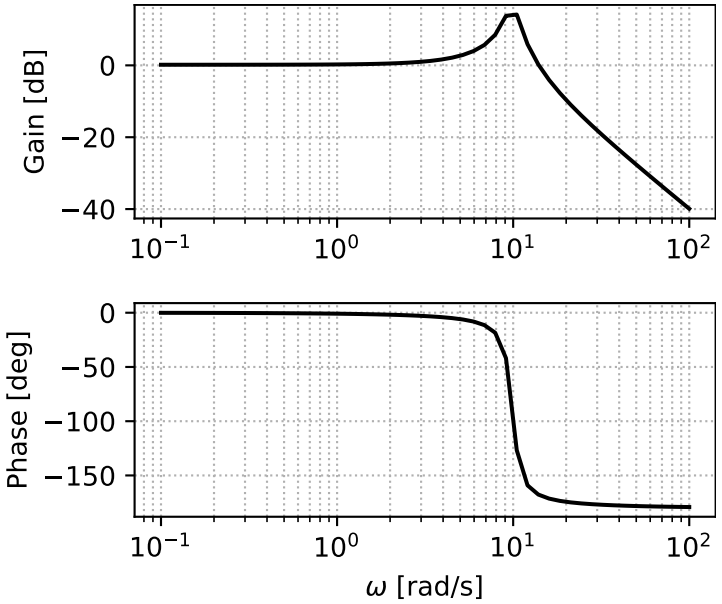
beta = 0.1
T2 = 0.1

ループ整形法の例題

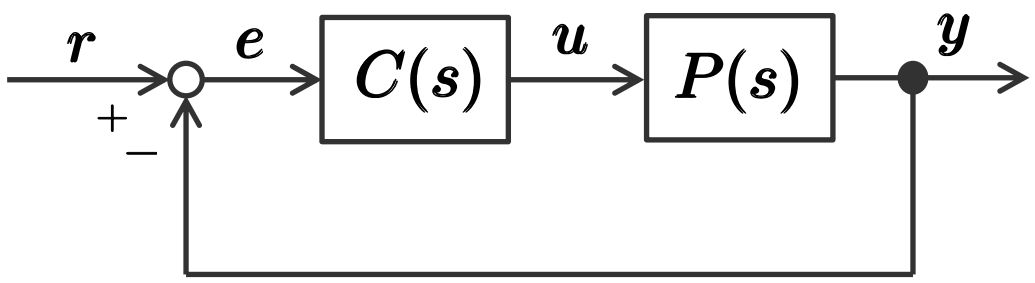


$$P(s) = \frac{1}{Js^2 + cs + mg\ell}$$

```
g = 9.81      # 重力加速度[m/s^2]
l = 0.2       # アームの長さ[m]
m = 0.5       # アームの質量[kg]
c = 1.5e-2    # 粘性摩擦係数
J = 1.0e-2    # 慣性モーメント
```



に対してフィードバック制御系を構成する



$$C(s) = kC_1(s)C_2(s)$$

ゲイン交差周波数を 40 rad/s
位相余裕を 60°
定常偏差を改善（定常偏差 0.02 以下）

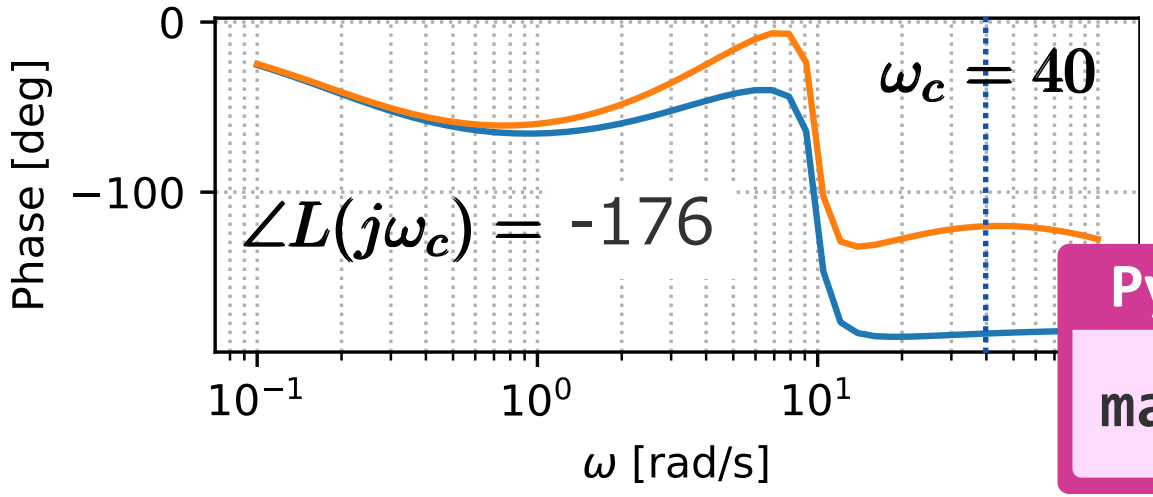
ループ整形法の例題：方針

1. 位相遅れを設計. $1/T_1$は, $\omega_c = 40$ より1/10の位置に選ぶ → $T_1 = 0.25$

   低周波ゲインを上げる→ $\alpha = 20$

2. 位相進みを設計. $\omega_c = 40$ のPMが60°になるようにする.

$L = PC_1$ の位相線図



$\phi_m =$ 60 -4 =56°

$$\beta = \frac{1-\sin\phi_m}{1+\sin\phi_m}$$

$$T_2 = \frac{1}{\omega_m\sqrt{\beta}}$$

Python：特定の周波数でのゲイン・位相の計算

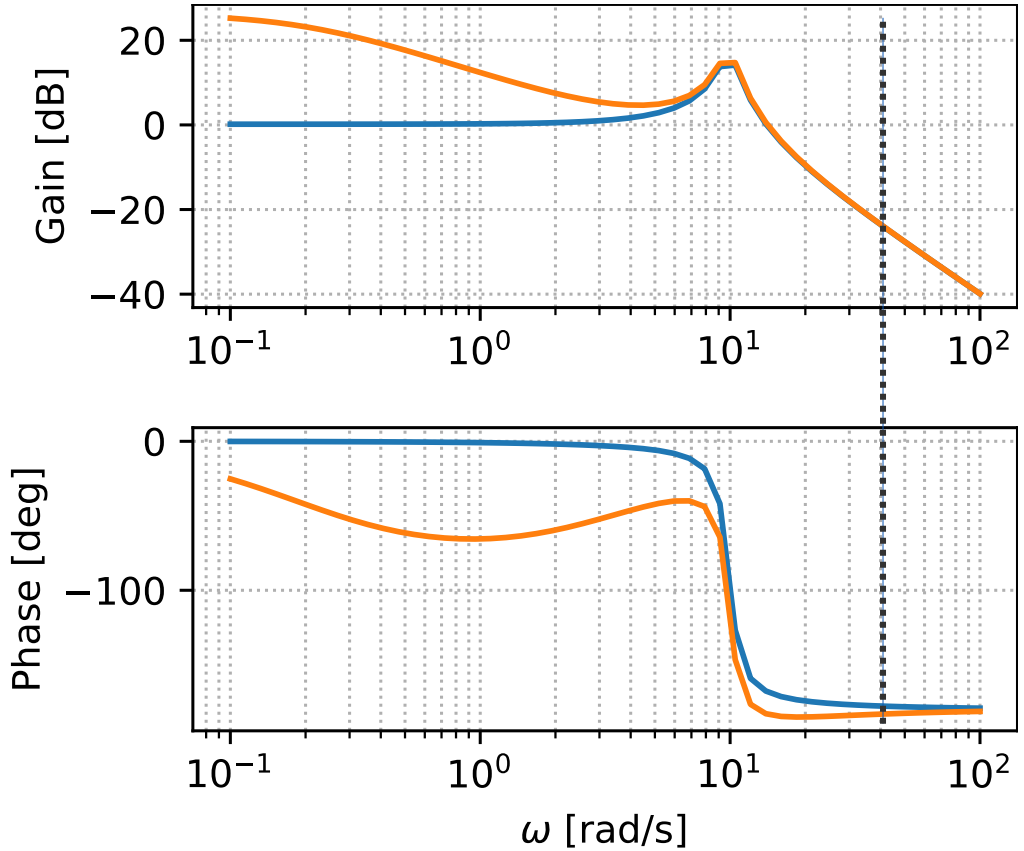
```
mag, ph, w = freqresp(モデル, [周波数])
```

3. ゲイン補償を設計. $L = PC_1C_2$ の $\omega_c = 40$ における
   ゲインが 0[dB] になるようにする.

ループ整形法の例題：位相遅れ補償


```
alpha = 20
T1 = 0.25
K1 = tf([alpha*T1, alpha], [alpha*T1, 1])
print('K1=', K1)

fig, ax = plt.subplots(2, 1, figsize=(4, 3.5))
gain, phase, w = bode(P, logspace(-1, 2), Plot=False)
ax[0].semilogx(w, 20*np.log10(gain))
ax[1].semilogx(w, phase*180/np.pi)

H1 = P*K1
gain, phase, w = bode(H1, logspace(-1, 2),
Plot=False)
ax[0].semilogx(w, 20*np.log10(gain))
ax[1].semilogx(w, phase*180/np.pi)

ax[0].grid(which="both", ls=':')
ax[1].grid(which="both", ls=':’)

[[[mag]]], [[[phase]]], omega = freqresp(H1, [40])
phaseH1at40 = phase * (180/np.pi)
print('-----------------------')
print('phase at 40rad/s =', phaseH1at40)
```

位相遅れ補償で低周波ゲインを上げる



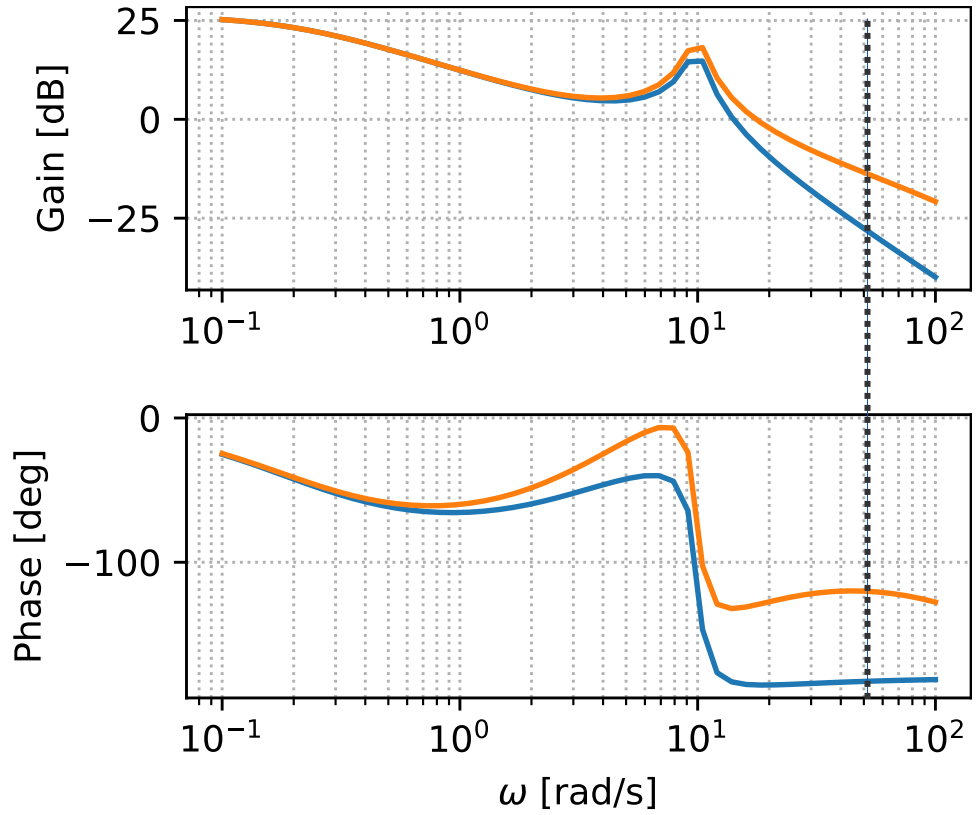
$$L = PC_1$$

40rad/sの位相　$\angle L(j\omega_c) =$ -176

ループ整形法の例題：位相進み補償


```
phim = (60- (180 + phaseH1at20 ) ) * np.pi/180
beta = (1-np.sin(phim))/(1+np.sin(phim))
T2 = 1/40/np.sqrt(beta)
K2 = tf([T2, 1],[beta*T2, 1])
print('K2=', K2)

fig, ax = plt.subplots(2, 1, figsize=(4, 3.5))

gain, phase, w = bode(H1, logspace(-1,2), Plot=False)
ax[0].semilogx(w, 20*np.log10(gain))
ax[1].semilogx(w, phase*180/np.pi)

H2 = P*K1*K2
gain, phase, w = bode(H2, logspace(-1,2), Plot=False)
ax[0].semilogx(w, 20*np.log10(gain))
ax[1].semilogx(w, phase*180/np.pi)

[[[mag]]], [[[phase]]], omega = freqresp(H2, [40])
magH2at40 = mag
phaseH2at40 = phase * (180/np.pi)
print('------------------------')
print('gain at 40rad/s =', 20*np.log10(magH2at40))
print('phase at 40rad/s =', phaseH2at40)
```

位相進み補償で40[rad/s]の位相を進ませる→ -120[deg]にする



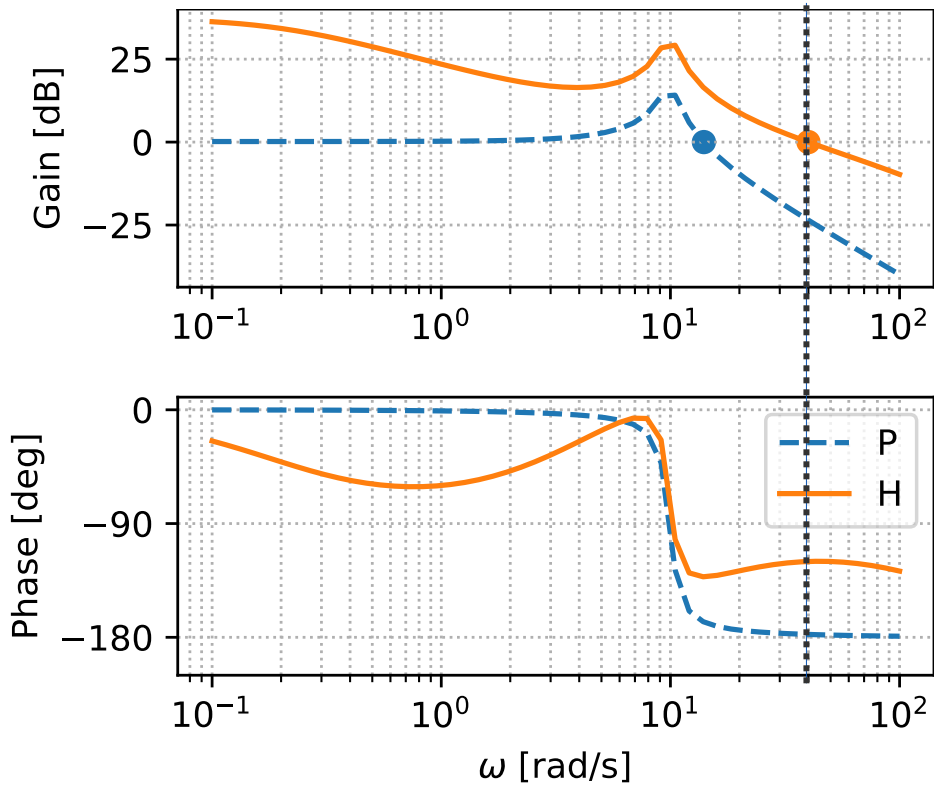
40[rad/s]でのゲインは-11.05 [dB]

ループ整形法の例題：ゲイン補償


```
fig, ax = plt.subplots(2, 1, figsize=(4, 3.5))

k = 1/magH2at40
print('k=', k)

gain, phase, w = bode(P, logspace(-1, 2), Plot=False)
ax[0].semilogx(w, 20*np.log10(gain), ls='--', label='P')
ax[1].semilogx(w, phase*180/np.pi, ls='--', label='P')

H = P*k*K1*K2
gain, phase, w = bode(H, logspace(-1, 2), Plot=False)
ax[0].semilogx(w, 20*np.log10(gain), label='H')
ax[1].semilogx(w, phase*180/np.pi, label='H’)
ax[1].legend()

print('-----------------')
print('(GM, PM, wpc, wgc)')
print(margin(H))
```

(GM, PM, wpc, wgc)
(inf, 60.000000000000003,
nan, 40.000000000000014)

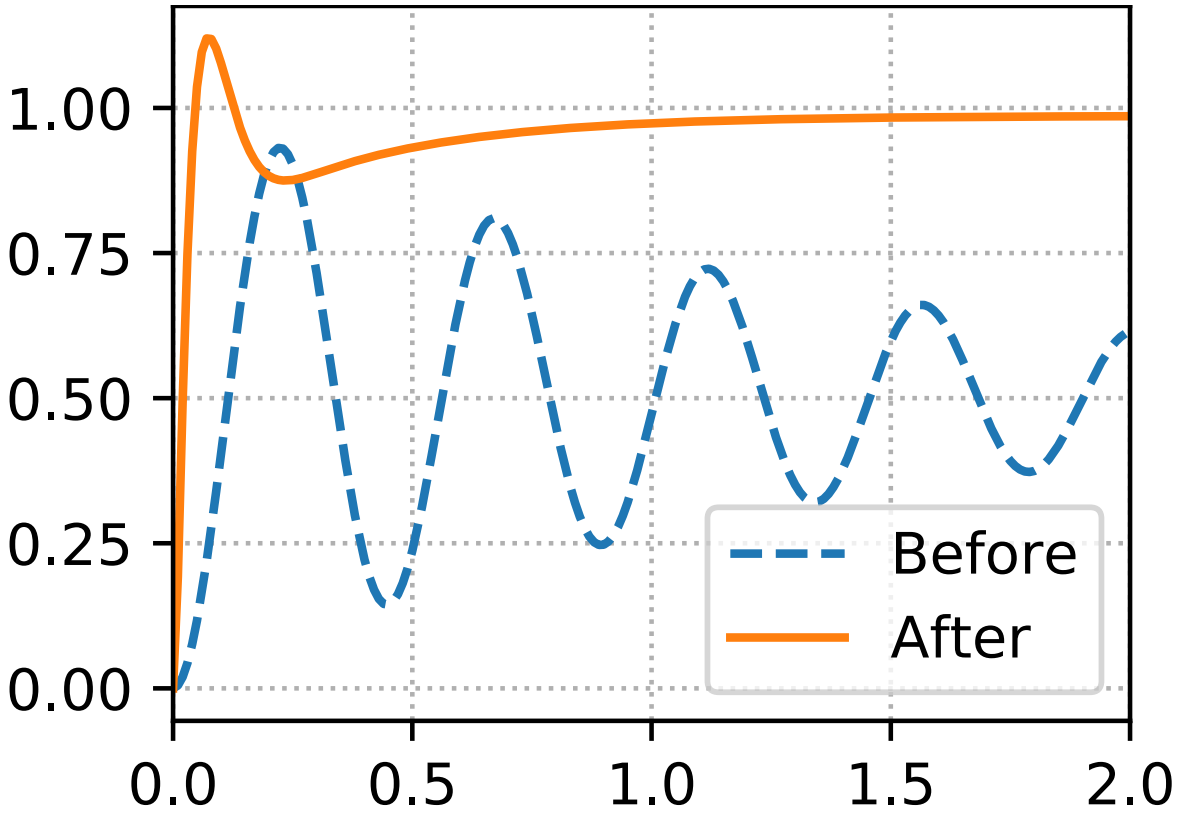
ゲイン補償で27.12[dB]上げる



ゲイン交差周波数を 40 rad/s
位相余裕を 60°
定常偏差を改善（定常偏差 0.02 以下）

ループ整形法の例題

閉ループ系のステップ応答

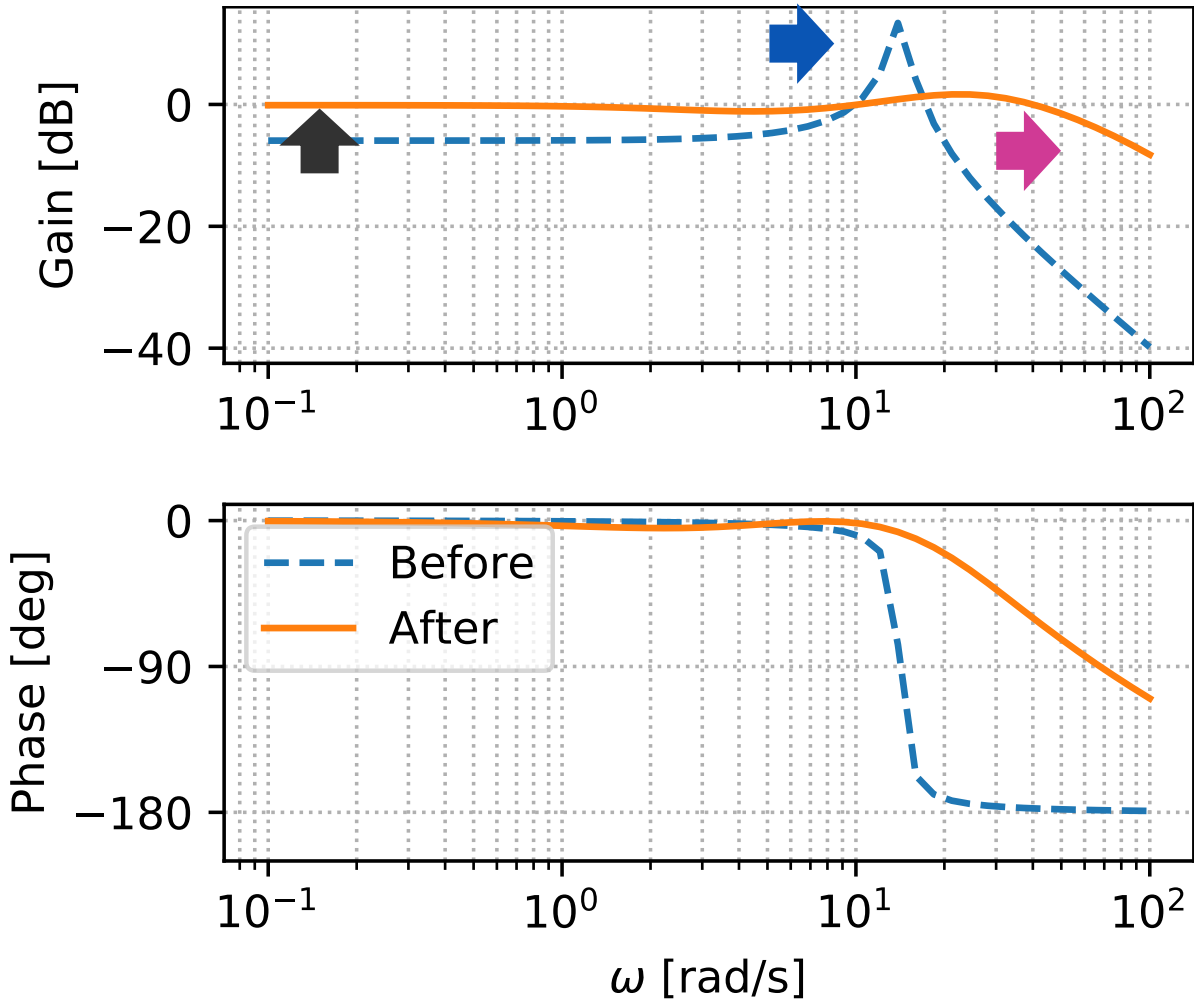


{'RiseTime': 0.031341050098324,
'SettlingTime': 0.848619202662311 3,
'SettlingMin': 0.874876349054517 9,
'SettlingMax': 1.121761604880379 6,
'Overshoot': 13.741171664614088,
'Undershoot': 0.0,
'Peak': 1.121761604880379 6,
'PeakTime': 0.074736350234464 91,
'SteadyStateValue': 0.986240592094559 7}

ゲイン交差周波数を 40 rad/s
位相余裕を 60°
定常偏差を改善（定常偏差 0.02 以下）

ループ整形法の例題

閉ループ系の周波数特性



閉ループ系の設計仕様

- バンド幅が十分大きい
- ピークゲイン $M_{\mathrm{p}} = \sup_{\omega} |G_{yr}(j\omega)|$ が小さい
- $|G_{yr}(0)| = 1$

勉強会の概要

Part 1

制御工学の基礎
- 制御とは，フィードバック制御，制御系設計

Pythonプログラミングの基礎
- Jupyter Notebook の使い方
- Pythonプログラミング　★ Python演習 1

制御系設計のためのモデル
- 伝達関数モデル，ブロック線図，時間応答，周波数応答　★ Python演習 2

Part 2

制御系設計のためのモデル（復習）
- 伝達関数モデル，時間応答，周波数応答

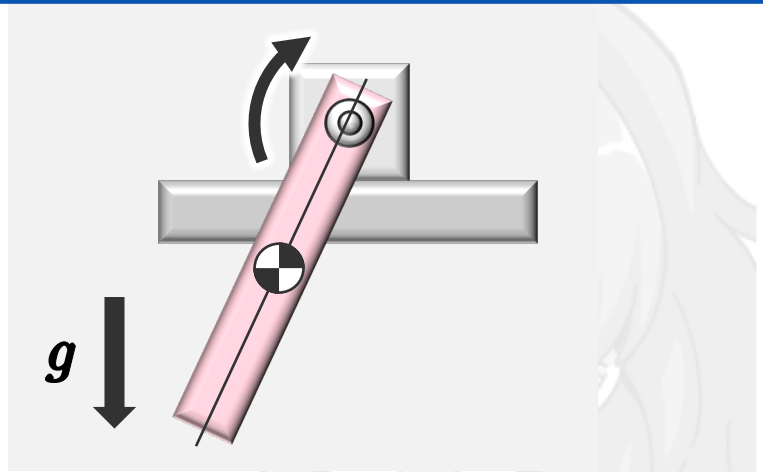
伝達関数モデルを用いた制御系設計
- 閉ループ系の設計仕様，PID制御，モデルマッチング　★ Python演習 3

ループ整形による制御系設計
- 開ループ系の設計仕様，位相進み・遅れ補償　★ Python演習 4

➡ **状態空間モデルを用いた制御系設計**
- 状態空間モデル，状態フィードバック　★ Python演習 5

姉妹のよくある日常会話

なんかできる子になったきがする



たしか，極配置．行列の固有値計算が面倒だった記憶がある

気のせいだとおもう笑
じゃつぎは，状態空間モデルを使う？

角度と角速度はセンサで取得できるから状態フィードバックにしよう

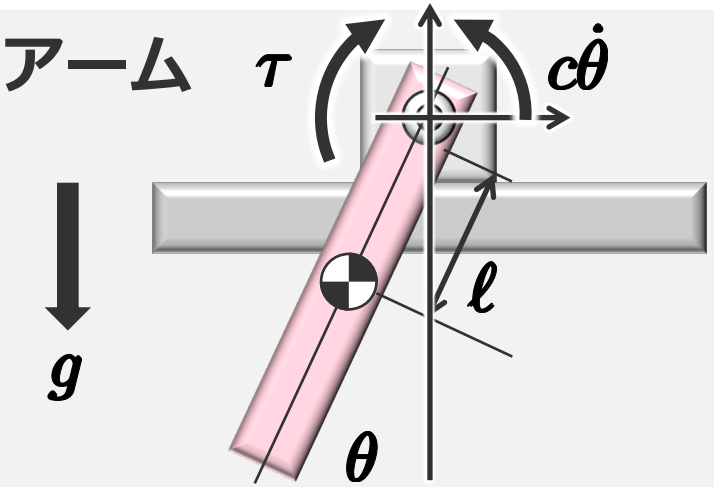
$$u = Kx$$

そう，それ．でも大丈夫，
Pythonが計算してくれるから

★まずは，状態フィードバック．その後，サーボ系やオブザーバ

機械系のモデル

垂直アーム



運動方程式

$$J\ddot{\theta}(t) = -c\dot{\theta}(t) - mg\ell\sin\theta(t) + \tau(t)$$

⬇ 線形化

$$J\ddot{\theta}(t) = -c\dot{\theta}(t) - mg\ell\theta(t) + \tau(t)$$

⬇ $y(t) = \theta(t) \quad u(t) = \tau(t)$

$$J\ddot{y}(t) + c\dot{y}(t) + mg\ell y(t) = u(t)$$

⬇ ラプラス変換

$$P(s) = \frac{y(s)}{u(s)} = \frac{1}{Js^2 + cs + mg\ell}$$

伝達関数

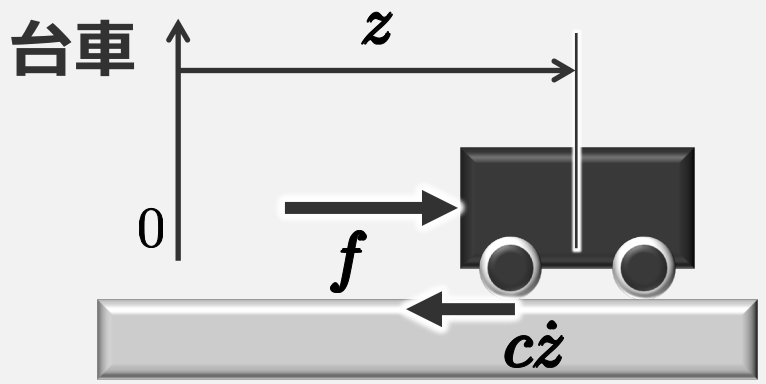
⬇ $x(t) = \begin{bmatrix}\theta(t)\\ \dot{\theta}(t)\end{bmatrix} \quad \begin{matrix}u(t) = \tau(t)\\ y(t) = \theta(t)\end{matrix}$

$$\dot{x}(t) = \begin{bmatrix}0 & 1\\ -\frac{mg\ell}{J} & -\frac{c}{J}\end{bmatrix} x(t) + \begin{bmatrix}0\\ \frac{1}{J}\end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix}1 & 0\end{bmatrix} x(t)$$

状態方程式

機械系のモデル



運動方程式

$$m\ddot{z}(t) = -c\dot{z}(t) + f(t)$$

状態の取り方には自由度がある

$$y(t) = \dot{z}(t) \quad u(t) = f(t)$$

⬇

$$m\dot{y}(t) + cy(t) = u(t)$$

⬇ ラプラス変換

$$P(s) = \frac{y(s)}{u(s)} = \frac{1}{ms + c}$$

伝達関数

$$y(t) = \dot{z}(t) \quad u(t) = f(t)$$

⬇

$$x(t) = \dot{z}(t)$$

⬇

$$x(t) = \begin{bmatrix} z(t) \\ \dot{z}(t) \end{bmatrix}$$

$$\dot{x}(t) = -\frac{c}{m}x(t) + \frac{1}{m}u(t)$$

$$y(t) = x(t)$$

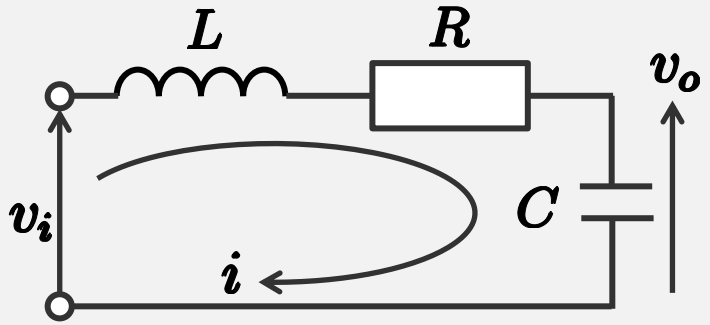
$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{c}{m} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(t)$$

状態方程式

電気系のモデル

RCL回路

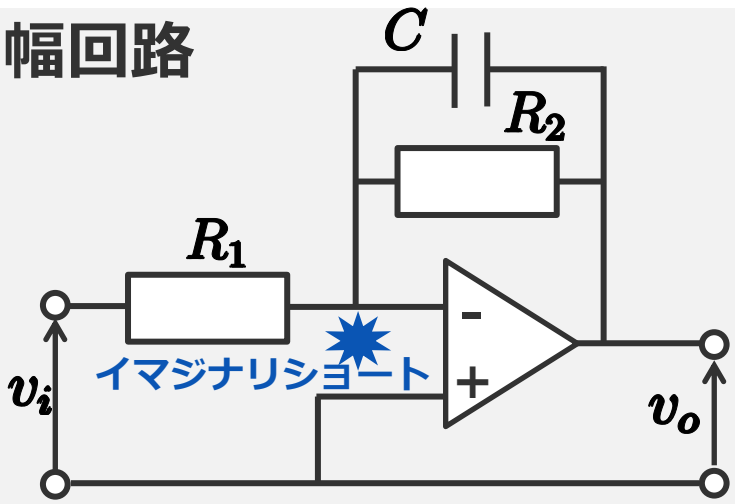


$$x(t) = \begin{bmatrix} \int i(\tau)\mathrm{d}\tau \\ i(t) \end{bmatrix} \qquad \begin{aligned} y(t) &= v_o(t) \\ u(t) &= v_i(t) \end{aligned}$$

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} \frac{1}{C} & 0 \end{bmatrix} x(t)$$

増幅回路



$$x(t) = v_o(t) \qquad \begin{aligned} y(t) &= v_o(t) \\ u(t) &= v_i(t) \end{aligned}$$

$$\dot{x}(t) = -\frac{1}{CR_2}x(t) - \frac{1}{CR_1}u(t)$$

$$y(t) = x(t)$$

伝達関数モデルと状態空間モデルの関係

状態空間モデル

$$P:\begin{cases}\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t)+\boldsymbol{B}u(t)\\ y(t) &= \boldsymbol{C}\boldsymbol{x}(t)\end{cases}$$

入力ー状態ー出力

一意に決まる

伝達関数モデル

$$P(s)=\frac{y(s)}{u(s)}=\boldsymbol{C}(sI-\boldsymbol{A})^{-1}\boldsymbol{B}$$

入力ー出力

無数にある（状態の取り方次第）

$$\boldsymbol{A}=\begin{bmatrix}0 & 1 & 0 & \cdots & 0\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ \vdots & & \ddots & \ddots & 0\\ 0 & \cdots & \cdots & 0 & 1\\ -a_0 & -a_1 & \cdots & \cdots & -a_{n-1}\end{bmatrix}\quad \boldsymbol{B}=\begin{bmatrix}0\\0\\\vdots\\0\\0\\1\end{bmatrix}$$

$$\boldsymbol{C}=\begin{bmatrix}b_0 & \cdots & b_m & 0 & \cdots & 0\end{bmatrix}$$

$$P(s)=\frac{b_m s^m+b_{m-1}s^{m-1}+\cdots+b_1 s+b_0}{s^n+a_{n-1}s^{n-1}+\cdots+a_1 s+a_0}$$

可制御正準形での実現（最小実現）

Python演習５

状態方程式のステップ応答

以下のコードを実行してみましょう

```
A = [[0, 1], [-4, -5]]
B = [[0], [1]]
C = [[1, 0], [0, 1]]
D = [[0], [0]]
```

Python ：状態空間モデルの定義

```
ss( A, B, C, D )
```

Python ：ステップ応答

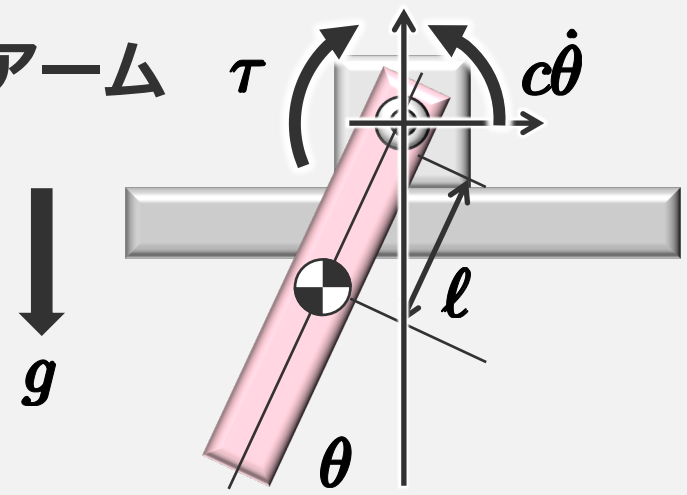
```
y, t = step(モデル, 時間)
```

出力 時間

```
from control.matlab import *
import matplotlib.pyplot as plt
import numpy as np

A = [[0, 1],[-4, -5]]
B = [[0], [1]]
C = np.eye(2)
D = np.zeros([2, 1])
P = ss(A, B, C, D)

Td = np.arange(0, 5, 0.01)
x, t = step(P, Td)

fig, ax = plt.subplots(figsize=(3, 2.3))
ax.plot(t, x[:,0], label = '$x_1$')
ax.plot(t, x[:,1], ls = '-.', label = '$x_2$')
ax.set_xticks(np.linspace(0, 5, 6))
ax.set_yticks(np.linspace(-0.4, 0.6, 6))
ax.legend()
ax.grid(ls=':')
```

状態空間モデルによる設計

垂直アーム



$$x(t) = \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \end{bmatrix} \qquad \dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -\frac{mg\ell}{J} & -\frac{c}{J} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

状態方程式 $\dot{x} = Ax + Bu$

$y = Cx$

行列指数関数 $e^{At} = \mathcal{L}^{-1}[(sI - A)^{-1}]$

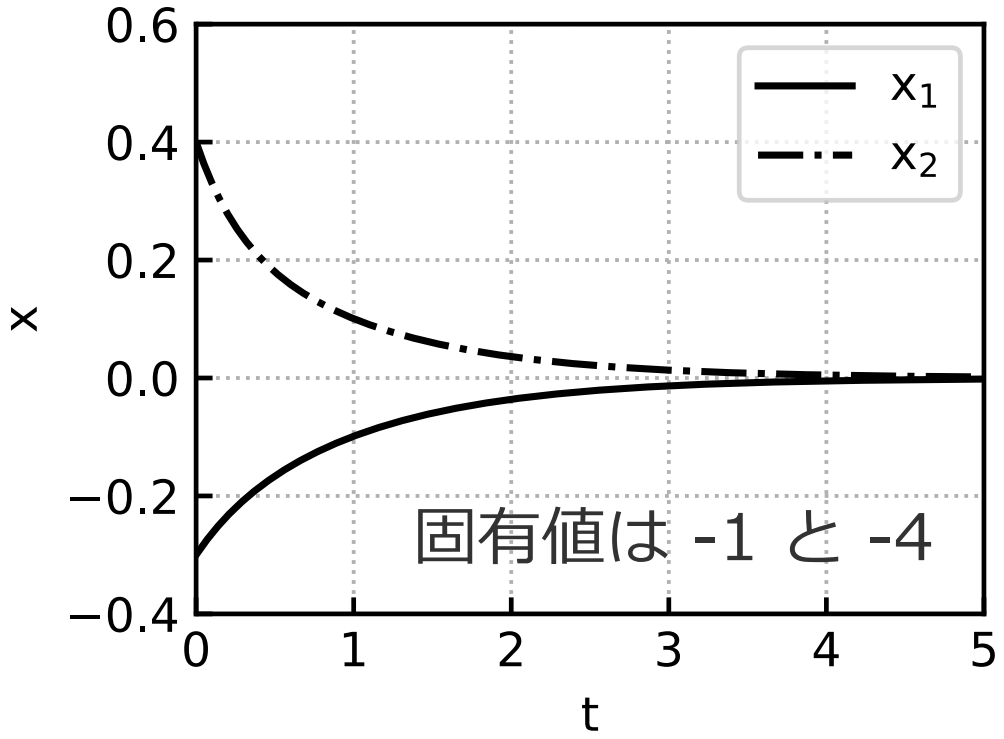
解の公式 $x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)\mathrm{d}\tau$

零入力応答　　零状態応答

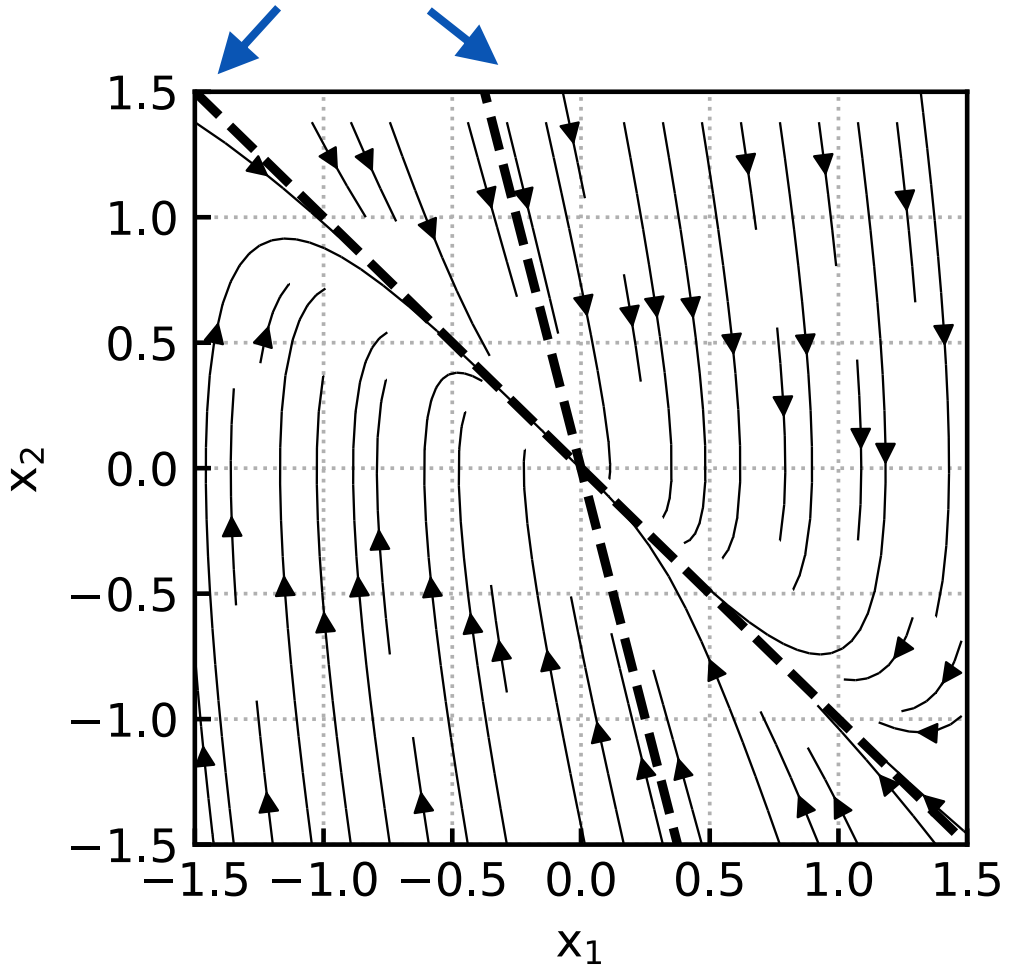
固有値と振る舞いの関係

- 初期値応答，位相面図

$$A = \begin{bmatrix} 0 & 1 \\ -4 & -5 \end{bmatrix} \qquad x(t) = e^{At}x(0)$$



-1と-4に対応する固有ベクトル

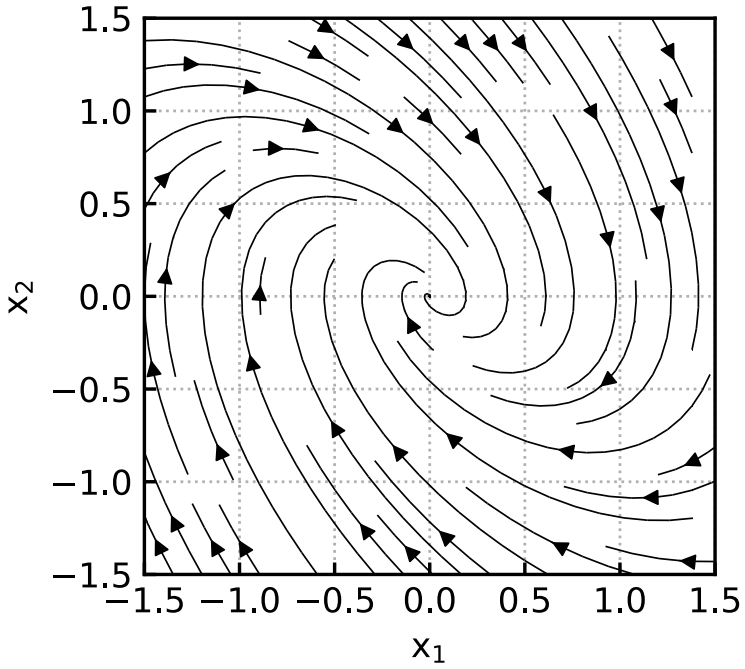


どの初期値からでも原点に収束する

固有値と振る舞いの関係


```
A = [[0, 1],
     [-1, -1]]
```

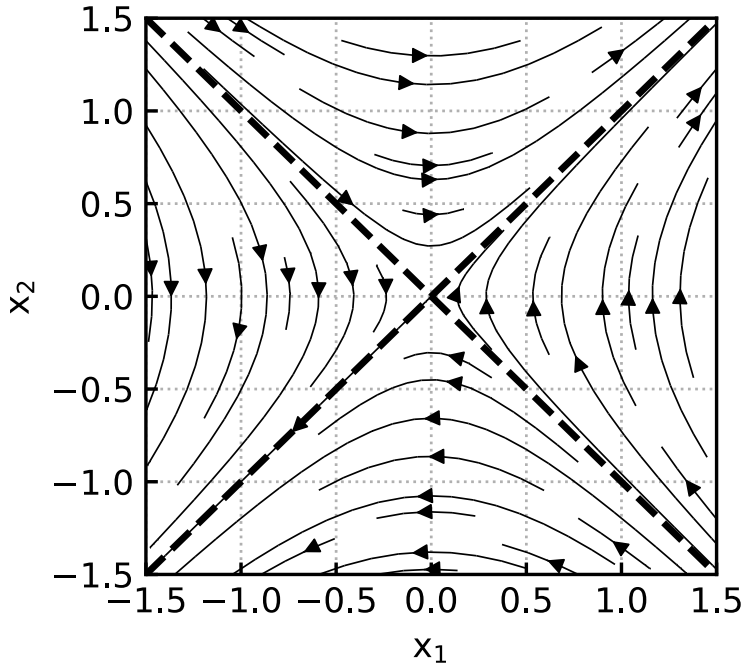
```
[-0.5+0.866j -0.5-0.866j]
```



振動しながら収束する

```
A = [[0, 1],
     [1, 0]]
```

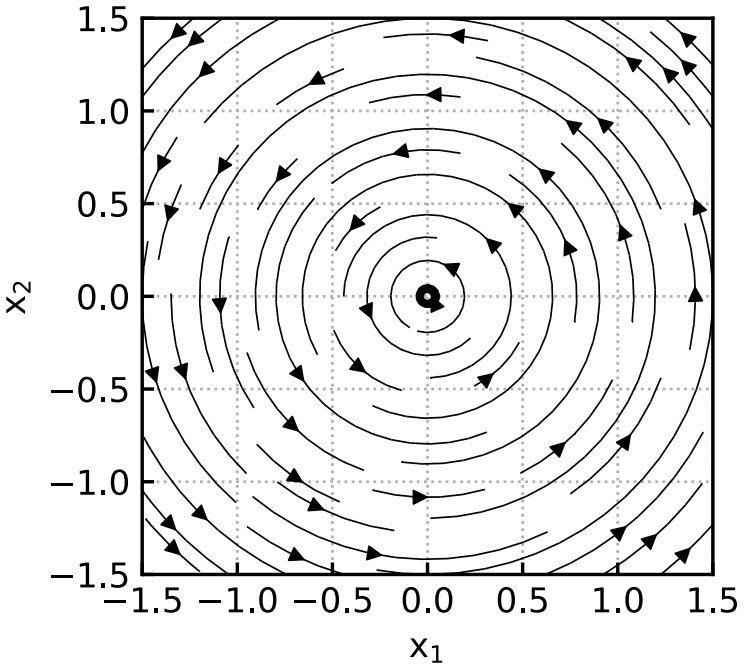
```
[ 1. -1.]
```



発散する

```
A = [[0, -1],
     [1, 0]]
```

```
[0.+1.j 0.-1.j]
```



振動しつづける

状態空間モデルの漸近安定性

$\dot{x} = Ax$ の応答は $x(t) = e^{At}x(0)$

行列指数関数 $e^{At} = \mathcal{L}^{-1}[(sI - A)^{-1}]$

の要素は「$e^{\lambda t}$ と t の多項式の積」の線形結合

λ は行列 A の固有値

A のすべての固有値の実部が負

$\longleftrightarrow$ $\dot{x} = Ax$ は安定（漸近安定）

Python ：固有値計算

```
np.linalg.eigvals(A)
```

λ が複素数だと，振動的な振る舞いになる

Python演習 5

状態空間モデルの応答

Python ：初期値応答

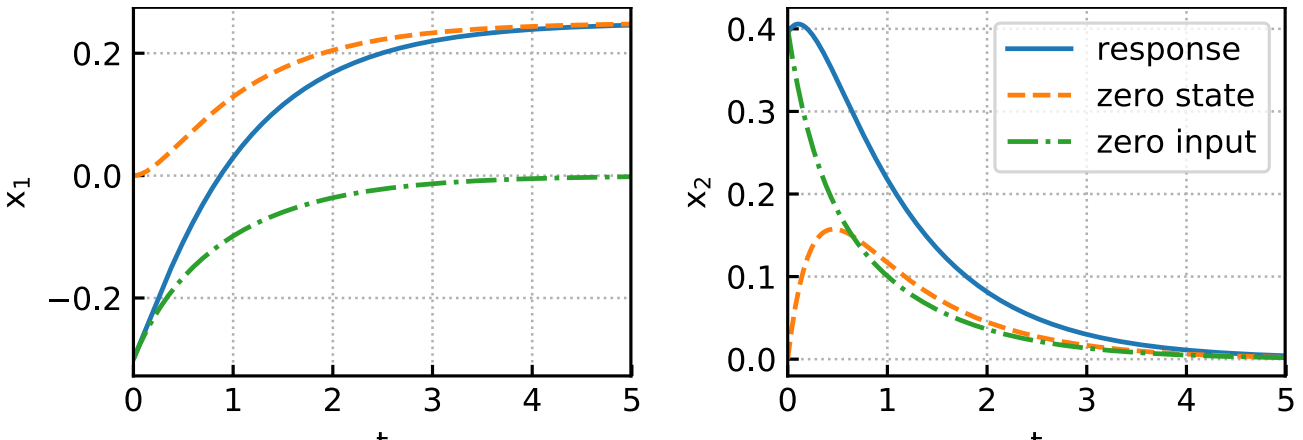
```
x, t = initial(モデル, 時間, 初期値)
```

状態 時間

Python ：時間応答

```
x, t, _ = lsim(モデル, 入力, 時間, 初期値)
```

状態 時間




```
A = [[0, 1],[-4, -5]]
B = [[0], [1]]
C = np.eye(2)
D = np.zeros([2, 1])
P = ss(A, B, C, D)

Td = np.arange(0, 5, 0.01)
Ud = 1*(Td>0) #ステップ入力
X0 = [-0.3, 0.4]

xst, t = step(P, Td) #ゼロ状態応答（ステップ入力）
xin, _ = initial(P, Td, X0) #ゼロ入力応答
x, _, _ = lsim(P, Ud, Td, X0)

fig, ax = plt.subplots(1, 2, figsize=(6, 2.3))

for i in [0, 1]:
    ax[i].plot(t, x[:,i], label='response’)
    ax[i].plot(t, xst[:,i], ls='--', label='zero state’)
    ax[i].plot(t, xin[:,i], ls='-.', label='zero input’)
    ax[i].grid(ls=':')

ax[0].set_xlabel('t')
ax[0].set_ylabel('$x_1$')
ax[1].set_xlabel('t')
ax[1].set_ylabel('$x_2$')
ax[1].legend()
fig.tight_layout()
```

状態フィードバック制御

レギュレータ問題

任意の初期状態に対して，状態変数 $x(t)$ を $x(t) \to 0 \quad (t \to \infty)$ とする

ただし，ここではシステムは可制御であるとする

状態フィードバック制御　　$\dot{x} = Ax + Bu$ を

$u = Kx$ で安定化する

方法１）　$A + BK$ の固有値を指定（実部が負）して，K を決める

➡ **極配置法**

方法２）　$J = \int_0^\infty (x^\mathsf{T} Q x + u^\mathsf{T} R u) dt$

を最小化する K を求める ➡ **最適レギュレータ（LQ）**

Python演習５：極配置

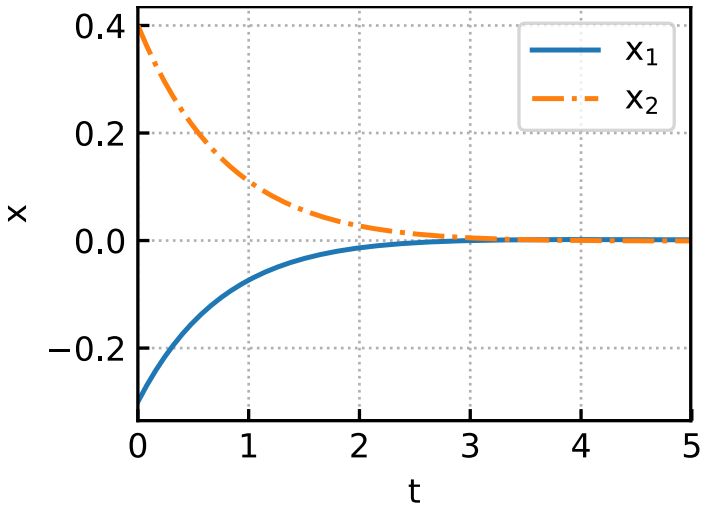

```
A = [[0, 1],[-4, 5]]
B = [[0], [1]]
C = np.eye(2)
D = np.zeros([2, 1])
P = ss(A, B, C, D)

Pole = [-1, -1]
F = -acker(P.A, P.B, Pole)

Acl = P.A + P.B*F
Pfb = ss(Acl, P.B, P.C, P.D)
Td = np.arange(0, 5, 0.01)
X0 = [-0.3, 0.4]
x, t = initial(Pfb, Td, X0) #ゼロ入力応答

fig, ax = plt.subplots(figsize=(3, 2.3))
ax.plot(t, x[:,0], label = '$x_1$')
ax.plot(t, x[:,1], ls = '-.', label = '$x_2$')
ax.grid(ls=':’)
```

Python：極配置

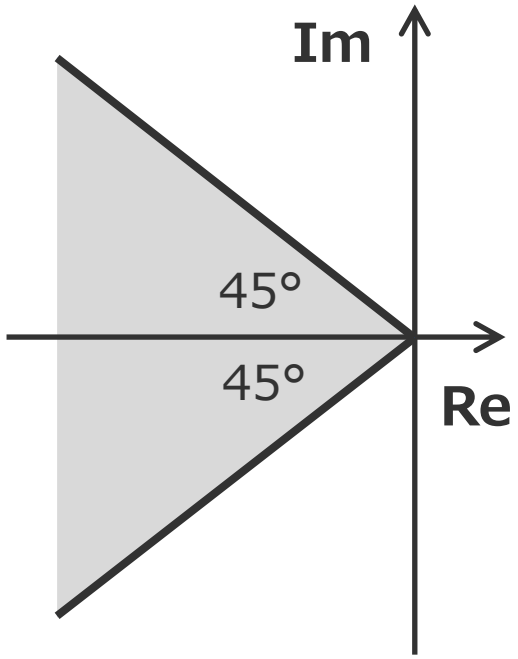
F = -acker(A, B, 指定極)

状態フィードバックゲイン

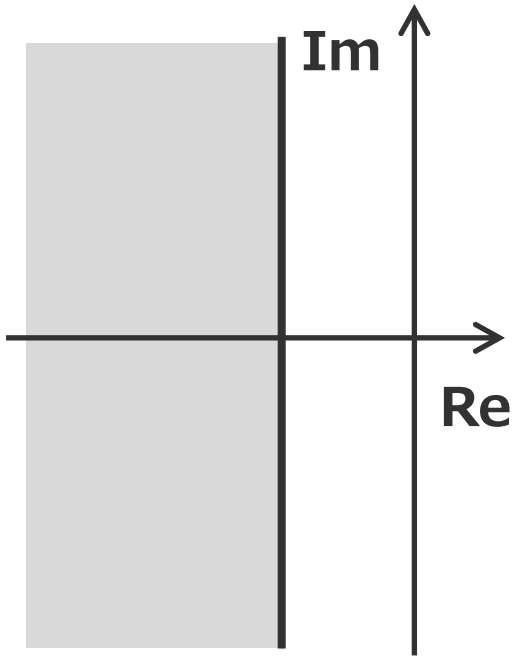
指定極（Pole）を変えて，
応答の違いを見てみよう



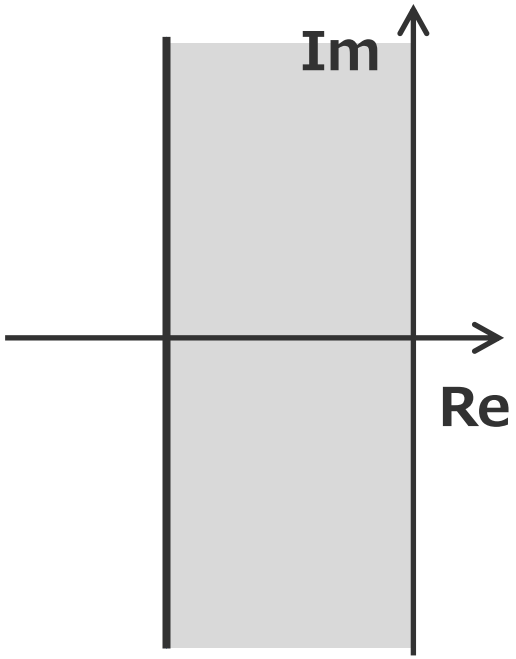
固有値をどこに配置するか？



減衰性の観点



速応性の観点



制御入力の観点

最適レギュレータ

レギュレータ問題（LQ）

$$J = \int_0^\infty (x^\mathsf{T} Q x + u^\mathsf{T} R u) dt$$

を最小化する K は　$K = -R^{-1} B P$

ただし，P　は，リカッチ方程式

$A^\mathsf{T} P + P A - P B R^{-1} B^\mathsf{T} P + Q = 0$ の正定対称解である

このときの閉ループ系の極は安定で，ハミルトン行列の安定固有値に等しい

$$H = \begin{bmatrix} A & -B R^{-1} B^\mathsf{T} \\ -Q & -A^\mathsf{T} \end{bmatrix}$$

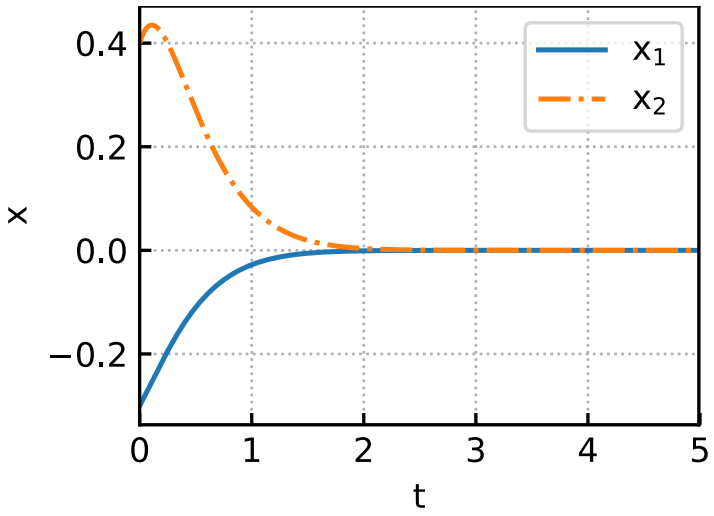
Python演習５：最適レギュレータ


```
Q = [ [100, 0], [0, 1]]
R = 1

F, _, _ = lqr(P.A, P.B, Q, R)
F = -F

Acl = P.A + P.B*F
Pfb = ss(Acl, P.B, P.C, P.D)

Td = np.arange(0, 5, 0.01)
X0 = [-0.3, 0.4]
x, t = initial(Pfb, Td, X0) #ゼロ入力応答

fig, ax = plt.subplots(figsize=(3, 2.3))
ax.plot(t, x[:,0], label = '$x_1$')
ax.plot(t, x[:,1], ls = '-.', label = '$x_2$')
ax.set_xlabel('t')
ax.set_ylabel('x')
ax.legend()
ax.grid(ls=':')
```

Python：最適レギュレータ

```
F, X, E = lqr(A, B, Q, R)
F = -F
```
状態フィードバックゲイン

重みQとRをを変えて，
応答の違いを見てみよう

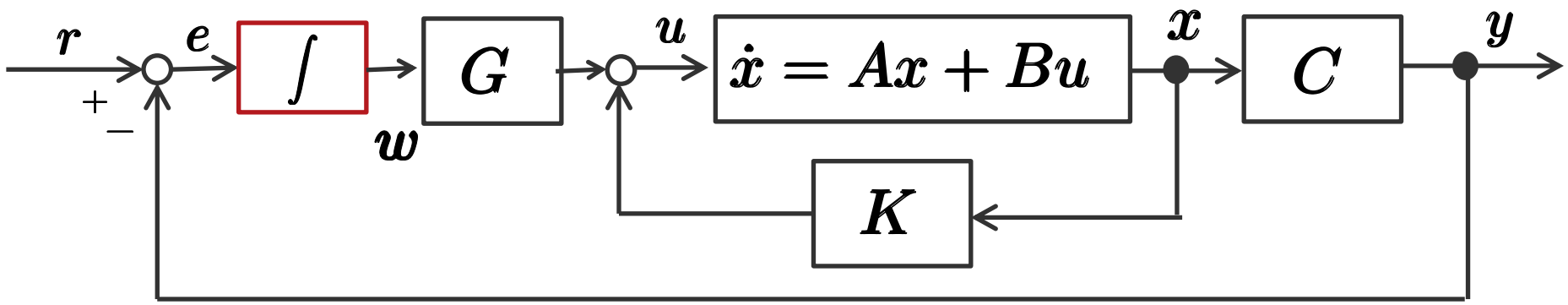


積分型サーボ系

サーボ問題

制御対象の出力 y が目標信号 r に定常偏差なく追従する

➡ **積分型サーボ系**　$u = Kx + Gw$



【内部モデル原理】

ステップ状の目標値に追従させる，
ステップ状の外乱を抑制する，
ためには，積分器が一つ必要

積分型サーボ系

$$u = Kx + Gw \qquad \dot{w} = r - y = r - Cx$$

$$\begin{bmatrix} \dot{x} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ I \end{bmatrix} r$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x_\infty \\ w_\infty \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_\infty + \begin{bmatrix} 0 \\ I \end{bmatrix} r$$

$$\tilde{x}_e := \begin{bmatrix} x - x_\infty \\ w - w_\infty \end{bmatrix}$$

$$\tilde{u}_e := u - u_\infty$$

$$\dot{\tilde{x}}_e = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \tilde{x}_e + \begin{bmatrix} B \\ 0 \end{bmatrix} \tilde{u}_e$$

$$\tilde{u}_e = \begin{bmatrix} K & G \end{bmatrix} \tilde{x}_e$$

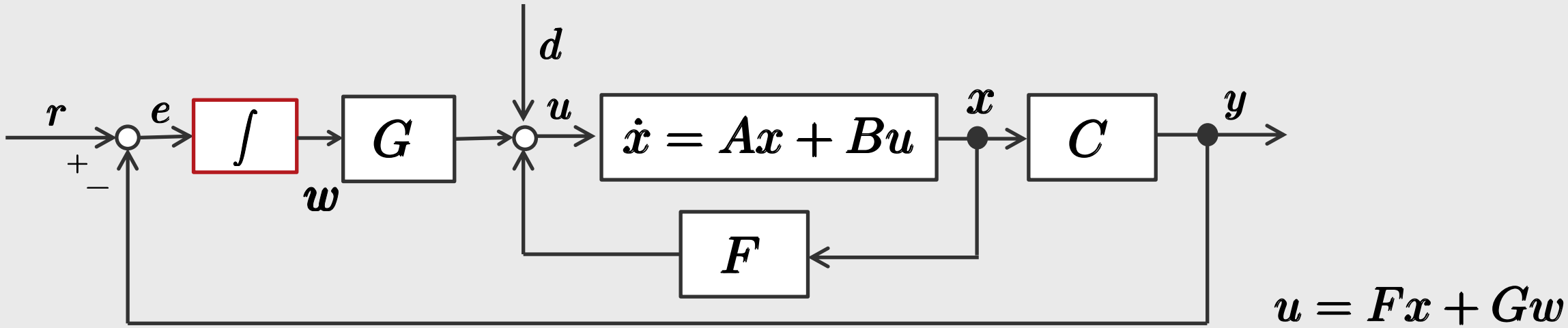
拡大系を用いて設計（極配置，最適レギュレータ）

サーボ問題の例題

制御対象 $\dot{x} = Ax + Bu$
$y = Cx$

$$A = \begin{bmatrix} 0 & 1 \\ -4 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

に対して１型サーボ系を構築する



➡ **拡大系** $\bar{A} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \quad \bar{B} = \begin{bmatrix} B \\ 0 \end{bmatrix} \quad \bar{C} = \begin{bmatrix} C & 0 \end{bmatrix} \quad K = \begin{bmatrix} F & G \end{bmatrix}$

サーボ問題の例題

最適レギュレータ問題（LQ）

$Q = Q^{\mathsf{T}} > 0 \quad R = R^{\mathsf{T}} > 0$ に対して，

評価関数 $J = \int_0^{\infty} \left(\bar{x}^{\mathsf{T}} Q \bar{x} + u^{\mathsf{T}} R u \right) \mathrm{d}t$

を最小化する $K = \begin{bmatrix} F & G \end{bmatrix}$ を求める

➡ $K = -R^{-1} \bar{B} P$

ただし，P は，リカッチ方程式

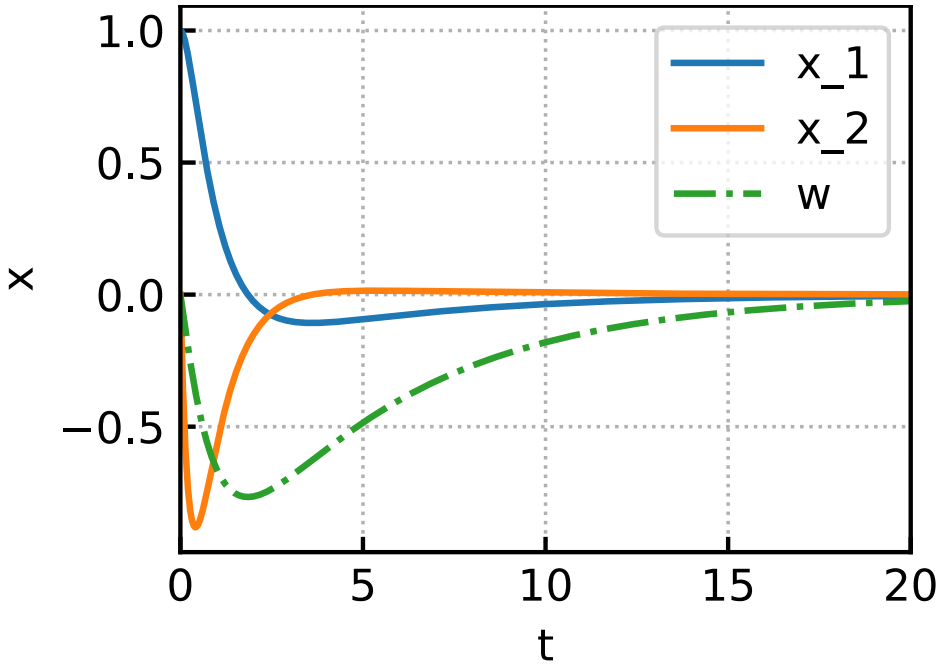
$\bar{A}^{\mathsf{T}} P + P \bar{A} - P \bar{B} R^{-1} \bar{B}^{\mathsf{T}} P + Q = 0$ の正定対称解である

Python：最適レギュレータ

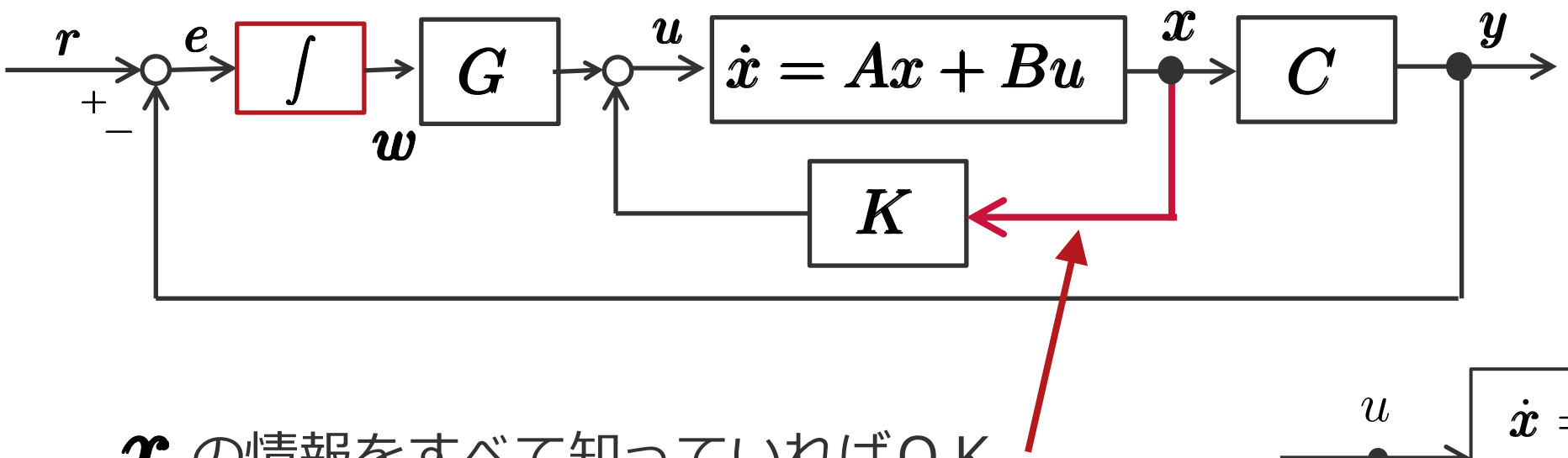
```
K, _, _ = lqr(A, B, Q, R)
K = -K  フィードバックゲイン
```

サーボ問題の例題


```
import numpy as np
from control.matlab import *

A = [[0, 1], [-4, 5]]
B = [[0], [1]]
C = [1, 0]
D = 0
P = ss(A, B, C, D)

Abar = np.r_[ np.c_[ P.A, np.zeros((2,1)) ],
              np.c_[ P.C, 0 ] ]
Bbar = np.c_[ P.B.T, 0 ].T
Cbar = np.c_[ P.C, 0 ]
x0 = np.array([[1], [0], [0]])

q1, q2, q3 = 10, 1, 1
Q = np.diag([q1, q2, q2])
R = 1
K, _, _ = lqr(Abar, Bbar, Q, R)
K = -K
```



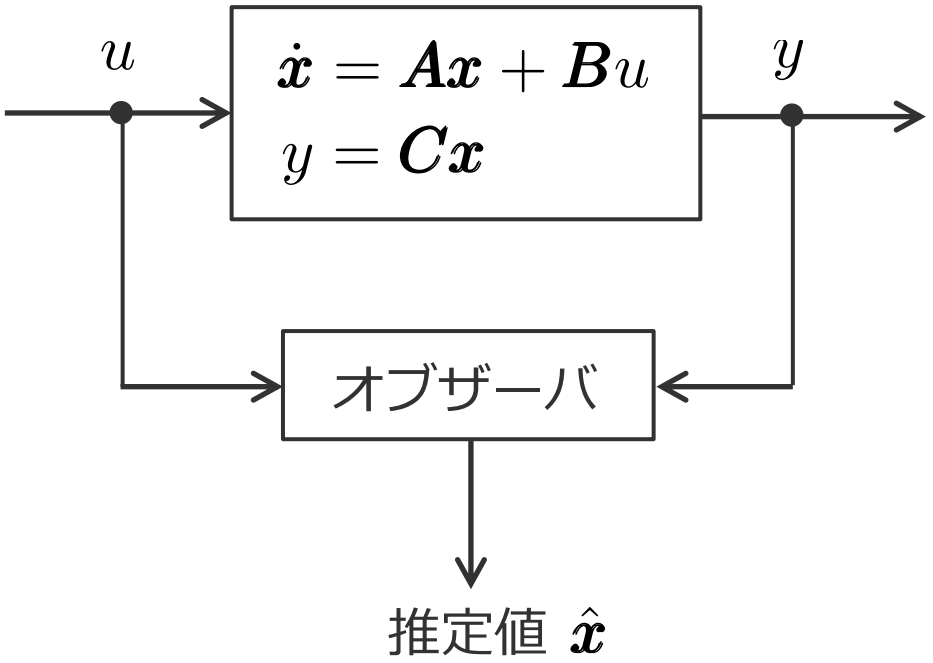
オブザーバ（状態推定）



$\boldsymbol{x}$ の情報をすべて知っていればＯＫ

⇒ 知らなければ，推定する必要がある

→ **オブザーバを用いて，出力 $\boldsymbol{y}$ と入力 $\boldsymbol{u}$ から $\boldsymbol{x}$ の推定値 $\hat{\boldsymbol{x}}$ を求める**



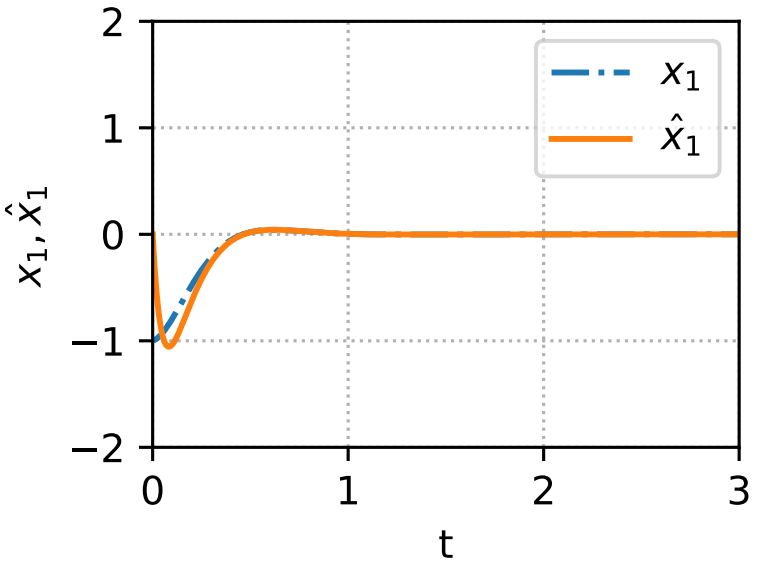
オブザーバ

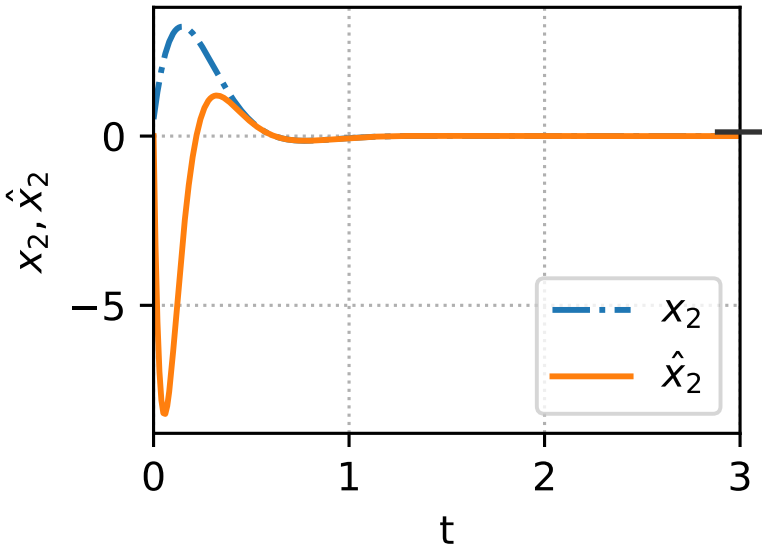
同一次元オブザーバ

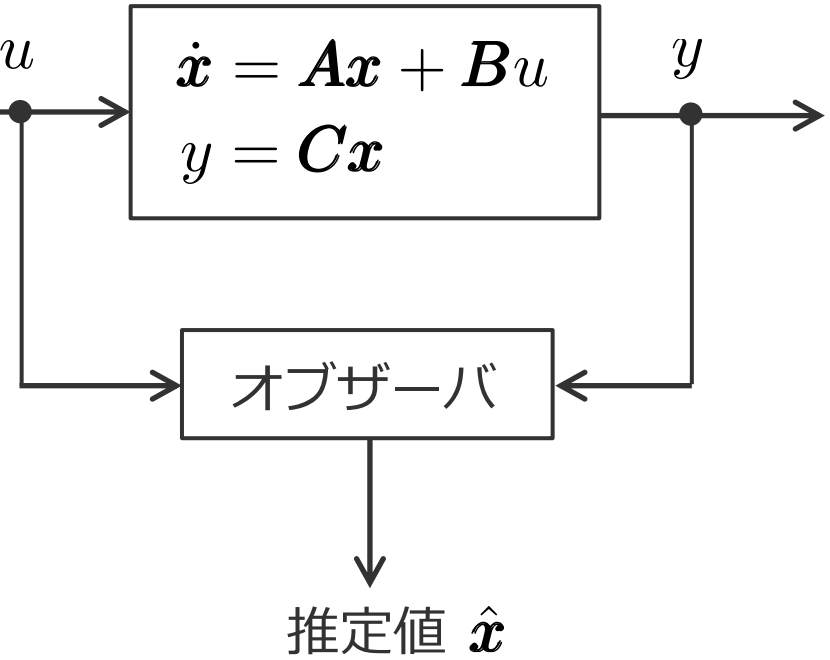
$$\dot{\hat{x}} = A\hat{x} + Bu - L(y - C\hat{x})$$

$$\dot{\hat{x}} = (A + LC)\hat{x} + \begin{bmatrix} B & -L \end{bmatrix} \begin{bmatrix} u \\ y \end{bmatrix}$$

L ：オブザーバゲイン







推定誤差　$e(t) = x(t) - \hat{x}(t)$　→　$\dot{e}(t) = (A + LC)e(t)$

オブザーバ

同一次元オブザーバ

$$\dot{\hat{x}} = A\hat{x} + Bu - L(y - C\hat{x})$$

$$\dot{\hat{x}} = (A + LC)\hat{x} + \begin{bmatrix} B & -L \end{bmatrix} \begin{bmatrix} u \\ y \end{bmatrix}$$

L ：オブザーバゲイン

設計方法）　$A + LC$ が安定（固有値の実部が負）となるように L を決める

$$\lambda(A + LC) = \lambda((A + LC)^{\mathsf{T}})$$

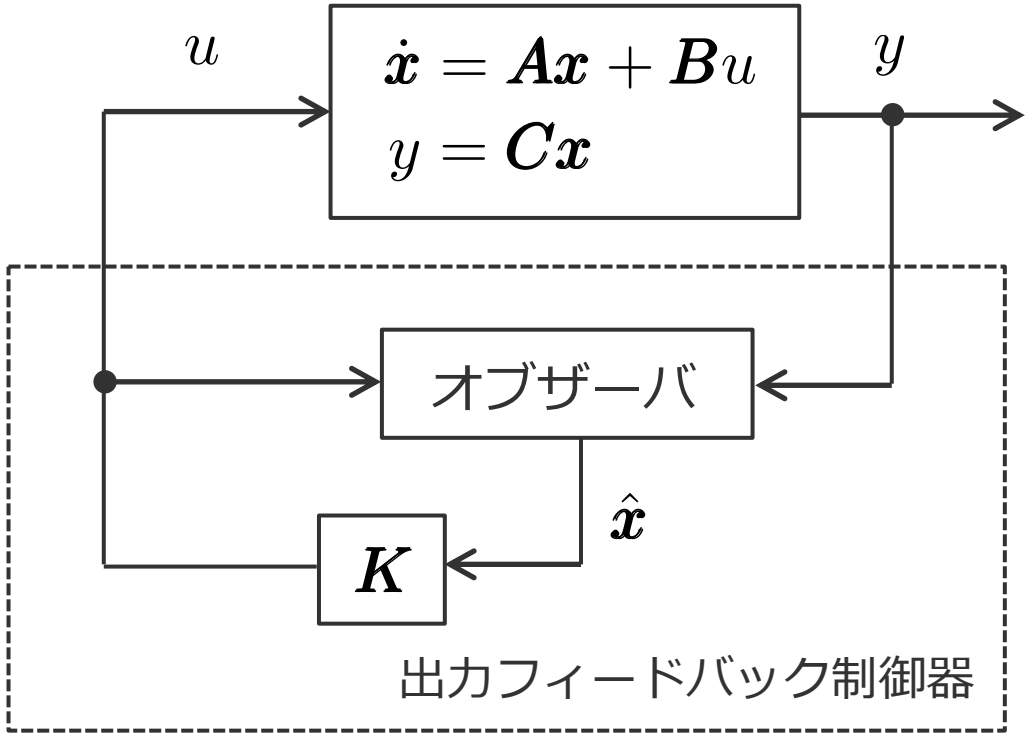
$$\longrightarrow \quad A^{\mathsf{T}} + C^{\mathsf{T}} L^{\mathsf{T}}$$

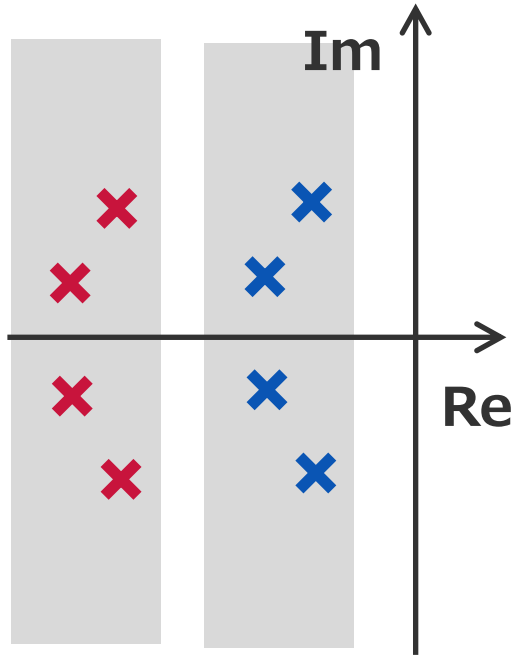
$A + BK$ レギュレータと同じ形

Python ：極配置

```
F = -acker(A.T, C.T, 指定極)
L = F.T                    オブザーバゲイン
```

オブザーバ





オブザーバ極　　レギュレータ極

$\boldsymbol{A} + \boldsymbol{LC}$　　$\boldsymbol{A} + \boldsymbol{BK}$

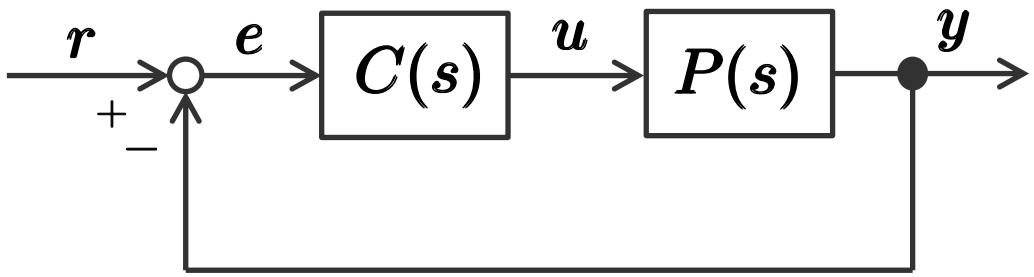
分離定理が成り立つので，

$\boldsymbol{u} = \boldsymbol{K}\hat{\boldsymbol{x}}$　としてOK（オブザーバとコントローラを独立に設計可能）

HomeWork

制御対象　$P(s) = \dfrac{10}{(s+1)(s+10)}$

に対してフィードバック制御系を構成する



$C(s) = kC_1(s)C_2(s)$　をループ整形により設計

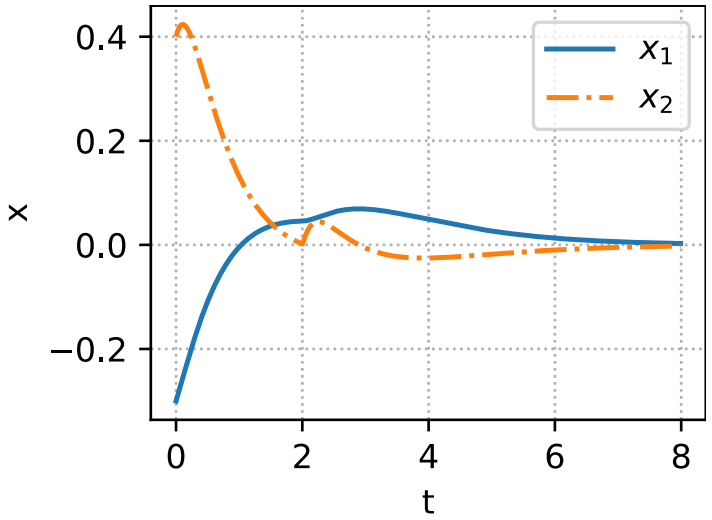
- ゲイン交差周波数を 20 rad/s
- 位相余裕を 60°
- 定常偏差を改善（定常偏差 0.01 以下）

HomeWork

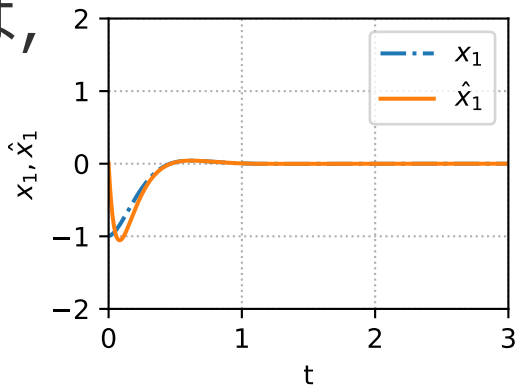
制御対象 $\dot{x} = Ax + Bu$, $y = Cx$

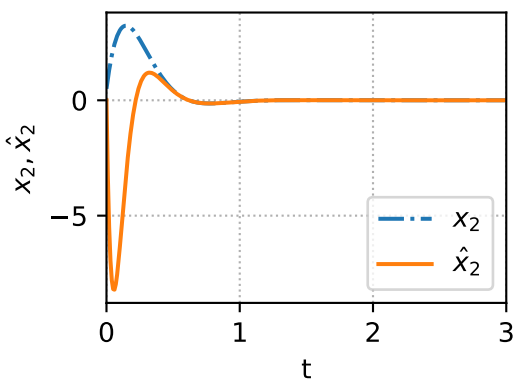
$$A = \begin{bmatrix} 0 & 1 \\ -4 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

に対して 1 型サーボ系を構築する
ただし，閉ループ極は -1, -1, -5 とする



上記の制御対象に対して，同一次元オブザーバを設計する．ただし，オブザーバ極は -15±5j とする．また，制御対象の安定化には，状態フィードバック制御を用いるものとし，レギュレータ極は -5±5jとする





今日の目標

Level☆☆★

- 開ループ系の設計仕様の用語を説明できる
- 開ループ系の安定余裕をPythonでチェックできる
- 状態空間モデルをPythonで記述できる

Level☆★★

- ループ整形法の流れが理解できる
- 極配置法で状態フィードバックゲインが設計できる

Level★★★

- 開ループ系の設計仕様を導くことができる
- 最適レギュレータやサーボ系，オブザーバを説明できる

次回予告？

システム制御情報学会・計測自動制御学会　チュートリアル講座 2020

Python で学ぶ制御系設計
―現代制御とアドバンスド制御―

申し込みはこちらから：https://www.iscie.or.jp/conf/meeting/tutorial/
定員 40 名、申し込み締め切り：5 月 26 日（火）

【日程】2020 年 6 月 4 日（木）　10:00～16:30

【場所】大阪工業大学 梅田キャンパス　（大阪府大阪市北区茶屋町 1 番 45 号）

主な交通アクセス

・JR「大阪」駅から徒歩 5 分　・阪急「大阪梅田」駅から徒歩 3 分　・阪神「大阪梅田」駅から徒歩 7 分

・御堂筋線「梅田」駅から徒歩 5 分　・谷町線「東梅田」駅から徒歩 5 分

（詳細は、https://www.oit.ac.jp/rd/access/index.html）

【講師】南　裕樹 氏（大阪大学）

【スケジュール】（休憩：12:00～13:00）

10:00 ～ 11:00　演習環境の準備と Python プログラミングの基礎（ハンズオントレーニング 1）

11:00 ～ 12:00　制御対象のモデルと動特性（ハンズオントレーニング 2）

13:00 ～ 14:30　現代制御理論による制御系設計（ハンズオントレーニング 3）

15:00 ～ 16:30　アドバンスド制御理論による制御系設計（ハンズオントレーニング 4）

現代制御のところを
もうすこし詳しく

さらに，LMIを用いた
設計まで？

乞うご期待！

各種仕様の推奨値

閉ループ系の仕様

	サーボ系	プロセス系
減衰係数	0.6～0.8	0.2～0.5
行き過ぎ量	0～25%	
ピークゲイン	1.1～1.5 (1.3が標準)	

開ループ系の仕様

	サーボ系	プロセス系
位相余裕	40°～60°	20°以上
ゲイン余裕	10～20dB	3～10dB

可制御性と可観測性

$$\dot{x} = Ax + Bu \quad y = Cx$$

可制御：任意の初期状態から，適当な時刻まで適当な入力を加えることで，状態を零とすることができるか？

$$\mathrm{rank}(M_c) = n \qquad M_c := \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}$$

```
Mc = ctrb (A, B)
```

可観測：有限時間区間の出力および入力から初期状態を一意に決定できるかどうか？

$$\mathrm{rank}(M_o) = n \qquad M_o := \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

```
Mo = obsv(A, C)
```